STUDY OF ONE- AND TWO-DIMENSIONAL FILTERING AND DECONVOLUTION ALGORITHMS FOR A STREAMING ARRAY COMPUTER

NASA GRANT NO. NSG-1648

FINAL REPORT

Appendix 2

Dr. George E. Ioup, Principal Investigator
Department of Physics
University of New Orleans
New Orleans, LA 70148

ALWAYS-CONVERGENT
ITERATIVE NOISE REMOVAL AND DECONVOLUTION
FOR IMAGE DATA

A Thesis

Submitted to the Graduate Faculty of the

University of New Orleans

in partial fulfillment of the

requirements for the degree of

Master of Science in Applied Physics

in

The Department of Physics

by

Mark A. Whitehorn

B.S., University of New Orleans, 1977

December, 1981

ACKNOWLEDGEMENTS

For his direction and guidance in this project, and for all his assistance in the preparation of this thesis, I am sincerely thankful to Dr. George Ioup. Also of great value to me in this work was the support provided by my wife, Kathleen, in programming and editing. For the services of the UNO Computer Research Center and the help provided by Jim Thomas I am very grateful, as this thesis would not have been the same without them. I would also like to express thanks to Dr. Terry Riemer and Dr. Joseph Murphy for the time and suggestions they provided.

Part of this work was funded by NASA grant NSG 1648.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Linear filtering techniques currently used for the restoration of noisy, blurred or otherwise degraded image data are discussed and new techniques related to the iterative techniques of Morrison and van Cittert are developed and implemented. Programs written for the implementation are discussed in the appendices. It is shown that the new techniques are convergent for any system response function, and they are applied to the task of restoring a severely blurred image.

A model of a linear shift-invariant optical system is constructed and used to generate synthetic data representative of the response of a simple optical instrument to various types of input. Noise generated by the instrument and by other phenomena associated with use of an optical system is characterized and added into the model output in various amounts to test its effects on subsequent data processing. Also included in the model is the effect of severe defocusing of the optics on the optical transfer function. Van Cittert's technique for deconvolution does not converge for the defocused system. Application of the new techniques for noise removal and deconvolution is made and it is shown that the results are extremely useful when both are applied together to noisy data.

Application of known non-linear time-domain constraints within the algorithm is discussed and tested. It is shown that the bandwidth of the data may actually be increased by applying these constraints. Results of processing noisy synthetic data indicate that the constraints are a very useful feature of the method. A comparison is made of the effects of using various combinations of constraints on several types of data, indicating that the rate of convergence is increased by the application of one or more constraints and that the most generally effective constraint for a given set of data will be determined by the character of the data. If the image consists of objects appearing on a black background, the most effective constraint should be the non-negativity constraint. The peak height limitation constraint will be most effective for objects with an extent greater than that of the impulse response of the system.

Introduction

One of the most common objectives in image processing is the removal of degradations such as those caused by atmospheric blurring, diffraction limited optics, and defocusing. This type of processing is termed image restoration. It is the objective of image restoration techniques to restore the image to the form it had before the degradations occurred. Useful results can include an increase in resolution and improved definition of the image. Restoration techniques rely on a mathematical description of the degradation and/or imaging system to apply the necessary corrections for constructing a truer image of the object. The term "object" here represents the real configuration of light sources comprising a self-luminous or illuminated object which is imaged by the optical system. The term "image" refers to the distribution of light intensities forming the output of the imaging system. The image will normally be a close representation of the corresponding object luminance distribution, but may suffer from various degradations such as those mentioned above plus the addition of noise.

This study is an application of a restoration technique which is related to the noise removal and deconvolution techniques of Morrison and van Cittert. It was formulated to remove a major drawback of the older techniques; they

failed to converge for a wide class of response functions. Both Morrison's noise removal and van Cittert's deconvolution techniques are iterative techniques applied in the time domain. They are based on the representation of a linear shift-invariant system as a convolution of the impulse response of the system with the input to the system. Morrison's iteration begins by first smoothing the system output to remove incompatible noise. The effect of the remaining iterations is to restore the data gradually to their initial state with only the incompatible noise removed. The effect of a finite number of iterations of Morrison smoothing is to perform a windowing operation on the transform of the data. The window is closely related to the transfer function of the system to provide a greater weight to components of the data with less attenuation and probably higher signal to noise ratios. Van Cittert's deconvolution technique begins by approximating the limiting solution $f_p(x,y)$ with the system output $g(x,y)$. The limiting solution, if the method converges for the impulse response $h(x,y)$, is $f_p(x,y)$. Intermediate iterations provide partially restored results which again have a spectrum weighted preferentially toward those components corresponding to larger values of the system transfer function $H(u,v)$.

Always-convergent versions of these two techniques for two-dimensional image data are developed in chapter two and implemented in chapter three. The advantage of these

techniques over inverse filtering is that they allow the flexible application of function domain constraints such as non-negativity of image data, peak height limitations, and finite extent of the image in the process of iterating toward the final solution. Since the constraints directly affect the spectrum of the result by sharpening edges in the image, it is possible to extend the transform of the result beyond the bandwidth of the system. The addition of constraints to the iteration makes the method non-linear, and no theoretical treatment of the method with constraints is attempted. Convergence requirements for the linear version (no constraints) and experimental results indicating the usefulness of the technique with and without constraints are presented in chapters two and four. Chapter three discusses the implementation of the method using digital techniques.

Chapter one presents a mathematical model of an optical imaging system and describes the generation of synthetic data from the model for processing. Several special functions are used in the first three chapters for convenience in describing optical systems and linear filtering processes. Following is a table of special functions and the notation used to describe an optical imaging system.

Notation and Special Functions

[Bracewell (1965), Frieden (1979)]

DFT	discrete Fourier transform
FFT	fast Fourier transform
OTF	optical transfer function
PSF	point spread function
SNR	signal to noise ratio

f(x,y)	object irradiance
g(x,y)	image irradiance
h(x,y)	point spread function
n(x,y)	random noise function

Capitalized function names
represent transforms of functions

$F\{f(x,y)\}=F(u,v)$;Fourier transform of function f(x,y)

$F\{g(x,y)\}=G(u,v)$

$F\{h(x,y)\}=H(u,v)$

$$\text{Rect}(x,y) = \begin{cases} 1 & ;\{|x|<1/2 \text{ and } |y|<1/2\} \\ 0 & ;\{|x|>1/2 \text{ or } |y|>1/2\} \end{cases}$$

$$\text{Rect}(r) = \begin{cases} 1 & ;|r|<1/2 \\ 0 & ;|r|>1/2 \end{cases}$$

$J_1(r)$ = 1st order Bessel function

$$\text{Sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

delta(x)= impulse symbol

$$\text{Shah}(x) = \sum_{n=-\infty}^{\infty} \text{delta}(x-n) \quad ;\text{for n an integer}$$

The integer N represents the number of sample points included in the discrete 1-D Fourier transform or across a square 2-D discrete Fourier transform.

The integers n_g and n_h represent the number of sample points across the square space domain sampling window used to record g(x,y) and h(x,y) respectively.

CHAPTER 1

OPTICAL IMAGING SYSTEM MODEL

Since this is a two-dimensional study with application to image data, a model of an optical imaging system is used to generate realistic synthetic data for input to the algorithm under test. The model selected is applicable to systems forming images of objects radiating spatially incoherent light. The assumption of spatial incoherence is nearly correct for most optical imaging situations. Further restrictions placed on the model itself will be elaborated below.

In order to calculate appropriate instrument response functions for application of noise removal and deconvolution, the model system is represented in terms of its optical transfer and point spread functions. The ideal point spread function (PSF) and optical transfer function (OTF) are computed and then degraded to represent a defocused system. The PSF $h(x,y)$ is defined to be the response of the system to a single object point radiator. The OTF is related to the PSF by a Fourier transformation; it represents the transfer function of the system in terms

of the spatial frequency components of the input.

Gaskill defines the (incoherent) diffraction image of a general two-element lens system [Gaskill (1978)]:

$$I(x,y)=I'(x,y)*h(x,y) \tag{1.1}$$

where I is the image plane irradiance, I′ is the irradiance of the geometrical image, and "*" implies convolution. Since Eq.(1.1) represents the output (the diffraction image) as the convolution of the geometrical image with the impulse response of the system, we may regard this portion of the optics as a linear shift-invariant system. This equation does not account for any differences between the object and geometrical image. Eq.(1.1) fails in the presence of aberrations such as coma which tend to cause the PSF to become shift-variant. Also, the shift-invariant property holds only within the limits of the field of view allowed by the Fresnel conditions, which require that the sum of object and image extents be small with respect to the distance between object and image planes [Gaskill (1978)].

It can be shown that the PSF h(x,y) may be written in terms of the aperture stop as

$$h(x,y) = a^2|P(ax,ay)|^2/(\text{area of aperture}) \tag{1.2}$$

where a is a constant related to the physical parameters of the system, and P(x,y) is the Fourier transform of p(x,y), the complex amplitude transmittance function of the aperture

stop. Using Eq.(1.2), h(x,y) may be computed for any given aperture function p(x,y). Choosing a clear circular aperture stop gives p(x,y)=Rect(r) and P(x,y) = F{Rect(r)} in two dimensions. The Fourier transform of the two-dimensional cylindrical Rect represents the complex amplitude response of the system to spatially coherent light if the stop is located at the Fourier transform plane of the system [Gaskill (1978)]. The squared magnitude of that response is the impulse response for incoherent light input.

For the purpose of constructing a model of a sampled data system, the PSF and OTF may be computed by means of the fast Fourier transform (FFT) in the above equation. The two-dimensional FFT used to implement this model is designed to maintain the origin of the transform near the center of the sampled data array (see appendix 1). The array size must be equal to a power of two for the simple one-dimensional algorithm used in this implementation and a choice of square data arrays 64 elements on a side is made for generation of the synthetic data.

The chosen aperture function, a circularly symmetric Rect, is sampled such that it has a radius of 16 sample points and then transformed into the function domain to get the coherent PSF. The squared magnitude of this PSF is then the incoherent PSF, and the OTF is computed simply by inverse transforming (-i transform goes from space domain to spatial frequency, +i transform performs inverse) the

incoherent PSF. By the convolution theorem, this operation is identical to performing a periodic convolution of the frequency domain aperture function with itself. A periodic convolution [Oppenheim and Schafer (1975)] is defined as the convolution of two replicated sequences such that there may be overlap of the two functions around the ends of the window defined by the finite extent of the sampled data. This replication is implicit in the finite Fourier domain representation of a sequence, since sampling at some finite interval 1/T in the frequency domain corresponds to replication in the time domain with interval $T=1/2s_c$. The relation of the coherent OTF (the aperture transmittance function) to the incoherent OTF is then given by the above convolution, and it is obvious that the OTF will no longer be flat. This will result in attenuation of high spatial frequency components of the input signal. It is of interest to note that the incoherent OTF, though not flat, has twice the non-zero width of the coherent OTF.

In order to represent a further degradation of the input signal beyond that introduced by the focused system, the OTF is modified to represent that of a severely defocused system. The blur OTF for a severely defocused lens is $H=J_1(ar)/(ar)$ [Goodman (1968)]. This OTF is multiplied by the OTF for the focused lens system to obtain the overall OTF for the degraded system. Actual computation was done by making use of the fact that the transform of $J_1(r)/r$ is just Rect(ar) [Hecht and Zajac (1974)].

Computation of the defocused PSF is then done in the function domain simply as a linear convolution of Rect(r) with the focused PSF. The radius of the defocusing Rect is 3 sample points, approximately twice the radius of the focused PSF resulting from the choice of a circular aperture with radius of 16 sample points in the frequency domain.

Perspective plots of the focused and defocused model PSF and OTF follow. Plots were done by the ASPEX program written at the Laboratory for Computer Graphics and Spatial Analysis at Harvard University. Each plot gives a perspective view of the 2-dimensional surface defined by the data array representing the function. The lines drawn are contours defining the amplitude of the PSF or OTF at the x,y coordinates associated with each sample point. Note that the defocused PSF shows a much wider maximum than the focused PSF and a slight depression at the origin. This corresponds to the defocused OTF being narrowed and given negative lobes from the multiplication by $J_1(r)/r$.

Since the problem of restoring an image without noise present is a relatively simple one, and since the occurrence of a noiseless image is rare, the system model must take into account the generation of noise in imaging equipment. There are several sources of noise in typical imaging systems. Electronic circuitry in photodetection apparatus such as photomultiplier and vidicon systems introduces shot and thermal noise to the system output. Film grain noise is

present in photographic systems, due to the random nature of the distribution of silver particles in the processed film. Quantization error may be regarded as noise introduced in digitizing an analog signal [Pratt (1978)].

Shot noise generated in a photomultiplier may be regarded as having an ordinate dependent gaussian amplitude distribution. This is to say that the variance of the noise is proportional to the signal amplitude. The actual dependence of the noise on the signal is [Billingsley (1979)]:

$$\text{std. dev.} = Q^{.5}$$

where Q represents the average number of photoelectrons released in a sampling period.

Film grain noise is dependent on the film granularity, which is a measure of the size of the silver grains in the film. The noise resulting from this granularity is also dependent on signal amplitude (transmittance of the film) with a proportionality:

$$\text{std. dev.}_T = \text{std. dev.}_{T1}(T(1-T)/T_1(1-T_1))^{.5}$$

where T_1 is a reference transmittance at which the granularity of the film is measured with a microdensitometer.

Thermal noise is one of the most common noise sources in electronic imaging systems. It is generated by random electron fluctuations in resistive elements of photodetectors and amplifiers. This type of noise may be

represented by an additive random Gaussian process with a zero mean and standard deviation independent of the input signal. Although this noise is negligible compared to shot noise in high gain detectors such as photomultipliers, it will likely dominate in regions of low signal amplitude.

Quantization noise results from the conversion of a continuous analog signal to a digital representation with a finite resolution, or number of bits. If one assumes that a fixed point binary representation is used, and that input values are rounded to the nearest quantization level within the range allowed by the digital number representation, the error due to quantization must be within the range of +/- one half of the quantization width. The quantization width is the smallest number which may be represented by the fixed point binary representation. Assuming that the sequence of errors e(n) represents a stationary random process, that the spectrum of the error sequence is flat (white noise), that there is no correlation between the signal and the error, and that the probability distribution of the error process is uniform over the range of possible error, the mean of the noise introduced will be zero, and the variance equal to the square of the quantization width divided by twelve [Oppenheim and Schafer (1975), Hamming (1962)]. The assumptions made above are valid for a complex signal (one with much structure) sampled with a quantization width which is small relative to the first difference of the signal.

Quantization noise may be assumed to be present in the model data due to the fact that they are represented by digital numbers in a fixed point format for input to the processing algorithms. The quantization width used is 10^{-3}, resulting in a variance of approximately 10^{-7}. Since the peak value of the synthetic g(x,y) is 137, the SNR of the quantization error is then 92 dB. This is the amount of noise on the "noiseless" g(x,y) used in the test runs of the restoration algorithm.

A second set of model data is used to test two lower SNR's. The model for thermal noise generation is used for this purpose. Program NOISE.FOR generates additive gaussian noise in a set of image data (see Appendix 2). The two signal-to-noise ratios chosen were 23 dB and 33 dB (200:1 and 2000:1). The lower SNR represents a high noise level, while the higher SNR is typical of a visually "clean" image [Andrews and Hunt (1977)].

CHAPTER 2

IMAGE RESTORATION TECHNIQUES

A linear shift invariant system may be modeled by a convolution operation:

$$g(x,y)=f(x,y)*h(x,y) \tag{2.1}$$

where g(x,y) is the output of the system, f(x,y) is the input, and h(x,y) is the impulse response of the system [Bracewell (1978)]. The aim of restoration is to eliminate the smoothing effect of h(x,y) on the output g(x,y).

One technique often used for restoration of images is the method of inverse filtering. If we define the principal solution f_p as having transform

$$F_p=G/H \;;\; |H|>0 \tag{2.2}$$
$$F_p=0 \;;\; H=0$$

then F_p will have been compensated for the attenuation caused by H and will look like Rect(r) assuming an optical system with a circular aperture and a delta function input [Frieden (1979)]. The bandwidth of the system is defined as the width of the band of frequencies for which |H|>0, and it

is obvious that F_p will lack any components in F which exceed the bandwidth of H. (For simplicity we are assuming that there are no interspersed zeroes within the OTF. This is true for any optical system having a clear aperture.) This implies that the resultant point spread function will be $J_1(r)/r$ [Hecht and Zajac (1975)]. The properties of this function will then define the resolution of f_p. This is not the highest possible resolution obtainable by linear filtering within the available bandwidth [Andrews and Hunt (1977)]. The sidelobes exhibited by this point spread function are relatively large (on the order of 10%) and alternating in sign, with the largest lobes being negative and nearest the central maximum. Since intensities in imagery must be non-negative, one would expect that this PSF is not the optimum, since one would desire to reduce the side lobes as much as possible and linear techniques exist to perform this function optimally.

Consideration of the effect of noise on inverse filtering also brings out difficulties in implementation of this method of image restoration. Allowing for additive noise in the model, Eq.(2.1) becomes:

$$g(x,y)=f(x,y)*h(x,y)+n(x,y) \qquad (2.3)$$

where n(x,y) represents noise present in the output of the instrument. Application of the inverse filtering technique to noisy data results in output:

$$F_p(u,v)=F(u,v)+N(u,v)/H(u,v) \; ; \; |H(u,v)|>0$$

The inverse filtering operation will then emphasize the noise wherever 1/H(u,v) is greater than 1. This will cause severe degradation of overall signal quality at frequencies for which the actual signal-to-noise ratio is low and H is small. It is possible to define an optimum bandwidth for inverse filtering [Frieden (1979)] based on minimization of the mean square error in f_p resulting from noise in g. For an OTF which decreases monotonically to zero as frequency approaches the cutoff point, and the assumption that the actual signal-to-noise ratio is constant for all frequencies within the bandwidth, the optimum bandwidth is determined by the frequency at which the modulus of the OTF equals the root noise to signal ratio. This indicates that knowledge of the noise is critical in optimum application of inverse filtering whenever the SNR is not high. Following is a discussion of more sophisticated techniques which eliminate the disadvantage of extreme noise sensitivity by allowing a partial restoration and effectively allowing the choice of a range of restored point spread functions logically related to the impulse response of the system.

Another method for restoring image data is van Cittert's iterative deconvolution technique. This method allows the use of powerful function domain constraints such as the non-negativity constraint for images, and the simultaneous application of other constraints such as peak

height limitations and finite extent. Also an advantage of van Cittert's iteration is the fact that it may be terminated short of the point that noise begins to render the result unsatisfactory. Van Cittert's iteration:

$$f_0(x,y)=g(x,y) \tag{2.4}$$
$$f_i(x,y)=f_{i-1}(x,y)+[g(x,y)-f_{i-1}(x,y)*h(x,y)]$$

defines the initial estimate of $f(x,y)$ to be the system output $g(x,y)$. The next approximation, $f_1(x,y)$, is taken to be the sum of f_0 and the convolution of h with the difference between g and the previous current estimate of f. This process continues with the application of constraints being made at each iteration. In the transform domain, the i^{th} iteration of van Cittert's (without constraints) may be represented as:

$$F_i=G\sum_{n=1}^{i}(1-H)^n$$

where the term multiplying G may be viewed as a window operating on G to produce F_i. It has been shown that the van Cittert iteration is convergent for $|1-H(u,v)|<1$. This places restrictions on the form of $h(x,y)$ for which the method may be used. One restriction is that the peak of the even part of $h(x,y)$ must lie at the origin. In the transform domain, $|1-H(u,v)|<1$ requires that $Re\{H(u,v)\}$ never become negative [Hill (1973), Hill and Ioup (1976)]. This requirement hinders the use of van Cittert's iteration on a severely defocused optical system due to the presence

of negative lobes in the OTF. A comparison of the behavior of van Cittert's technique with that of the always-convergent iterative technique for h(x,y)=Rect(x,y) was presented by the author in a paper delivered at the 1981 meeting of the Louisiana Academy of Sciences. The effect of the divergence, if slight, is to reduce the amount of restoration possible before the divergence significantly affects the result [Ioup (1979)]. Further drawbacks to linear versions of this technique are the facts that for moderate values of i and small values of H(u,v) the result of the iteration without constraints is a linear version (i+1)G(u,v) of the input G(u,v) plus a linearly enhanced version (i+1)N(u,v) of the noise [Frieden (1979)].

Since the presence of even small amounts of noise in g(x,y) destroys the usefulness of inverse filtering techniques, it is necessary to perform a noise removal or attenuation operation prior to the application of any deconvolution technique related to inverse filtering. Morrison's iterative noise removal technique has been successfully used with the application of van Cittert's deconvolution technique [Ioup (1968)]. Morrison smoothing is defined as an iteration in the function domain:

$$g_0(x,y)=0 \tag{2.6}$$

$$g_i(x,y)=g_{i-1}(x,y)+[g(x,y)-g_{i-1}(x,y)]*h(x,y)$$

where $g_i(x,y)$ is the output from the i^{th} iteration of the algorithm. The transform domain representation of the i^{th}

iteration is:

$$G_i(u,v)=G(u,v)\{1-[1-H(u,v)]^i\} \qquad (2.7)$$

and the output may be regarded as the result of performing a windowing operation upon G(u,v) in the transform domain. The effects of this window are to remove incompatible noise from g(x,y) and to attenuate all components of g(x,y) for which H(u,v) is small [Ioup (1968), Morrison (1963)]. Incompatible noise is any noise having spectral components nonexistent in H(u,v) and compatible noise is that portion of the noise present with spectral components within the bandwidth of the OTF H(u,v). The de-emphasis provided by this noise removal technique where H(u,v) is small has the effect of reducing the magnitude of signal and noise in regions of G(u,v) corresponding to small H(u,v). This de-emphasis provides a reasonable compromise of resolution for decreased noise sensitivity since we know that the optimum bandwidth for linear filtering is related to the actual SNR as a function of frequency and the SNR may be assumed to be lowest for spectral components associated with small H(u,v) and additive white noise. Figure 2.1 is a perspective plot of the window defined by Morrison smoothing as a function of iteration number for a particular one dimensional G(s). The iteration number, n, increases from 1 at the front to 63 at the back, and the gradual change in shape of the window from G(s) to Rect(as) is apparent as n increases.

A new method proposed by Ioup is implemented here in two dimensions for the restoration of blurred image data and to take advantage of the iterative application of constraints in the space domain for extrapolation of the input signal transform beyond the bandwidth of the OTF. The method is similar to inverse filtering in that an estimate of f(x,y) is made in the same way that $f_p(x,y)$ is defined above. The method is also iterative to allow the gradual application of constraints to the data as restoration is performed. Also implemented is a modified version of Morrison's iterative smoothing technique for reducing the effects of noise on the iteration [Ioup (1979)]. In order to assure convergence, a new windowing function is defined in the transform domain by normalizing $H_m(u,v)$ to have a maximum amplitude of one:

$$H_m(u,v)=|H(u,v)|/|H_{max}(u,v)| \tag{2.8}$$

The deconvolution iteration is:

$$f_0(x,y)=g(x,y) \tag{2.9}$$
$$f_i(x,y)=f'_{i-1}(x,y)+[f_p(x,y)-f'_{i-1}(x,y)]*h_m(x,y)$$
$$f'_{i-1}(x,y)=\text{constrained}\{f_{i-1}(x,y)\}$$

Convergence of this iteration is assured if, in the transform domain, $|1-H_m(u,v)|<1$. Due to the definition of $h_m(x,y)$ convergence is assured for any h(x,y) normalized by Equation (2.8). If h(x,y) is non-negative, then

normalization is done by forcing it to have unit area, assuring a peak transform value of 1. The use of $f'_{i-1}(x,y)$ as the previous value for $f_{i-1}(x,y)$ allows the application of constraints to affect both the limit of resolution possible with the method and the noise sensitivity of the method. The addition of constraints to the process of restoring $g(x,y)$ to $f_p(x,y)$ makes the method non-linear, increasing the difficulty of analytically describing the results. This study simply applies the method to synthetic data with and without constraints in order to get a qualitative measure of the effectiveness of the method in the presence of varying amounts of noise.

Since this implementation of the convergent technique utilizes transform domain convolutions, it is possible to contemplate performing non-integral and multiple numbers of iterations in one step. This is made possible by defining a recursion in the transform domain:

$$F_i(u,v)=F_p(u,v)-[F_p(u,v)-G(u,v)][1-H_m(u,v)]^i \qquad (3.4)$$
$$F_{2i}(u,v)=F_p(u,v)-[F_p(u,v)-F_i(u,v)][1-H_m(u,v)]^i$$

where i is any real number. The result of performing this recursion n times is:

$$F_{ni}(u,v)=F_p(u,v)-[F_p(u,v)-G(u,v)][1-H_m(u,v)]^{ni} \qquad (3.5)$$

It is possible to work in the transform domain with this

recursion, performing any equivalent number of time domain iterations in one step before returning to the function domain to apply constraints. Input parameters to the program determine the number of equivalent iterations to be performed and the interval at which constraints will be applied.

Application of the non-negativity and finite extent constraints is simply a matter of setting to zero any output points which violate the constraint. In order to apply the peak constraint, however, one must have some knowledge concerning the signal input to the instrument. If it is possible to say that the input signal could not exceed some upper limit in magnitude, then convergence and resolution may be aided by application of a peak constraint between iterations. Since computation of F_p is done as a division by H, it is possible that the magnitude of $F_p(0,0)$ will have been changed with respect to the magnitude of G(0,0). This is a result of our having placed no restrictions on the area of h(x,y). If the area of h(x,y) is not unity the calculated f_p will have been corrected for the amplitude scaling to which this corresponds in the instrument. In order to use the above recursion with a peak constraint it is necessary to correct g(x,y) for this scaling. Dividing g(x,y) by the area of h(x,y) ensures the proper correspondence between the original f(x,y) (input to the instrument) and g(x,y) for application of peak constraints. Note that without the application of any constraints the

limit of the iteration is $f_p(x,y)$ regardless of whether $g(x,y)$ is scaled prior to starting, although the rate of convergence may be changed by the scaling.

CHAPTER 3

IMPLEMENTATION OF CONVERGENT ITERATIVE TECHNIQUES

Implementation of the convergent iterative techniques for noise removal and deconvolution requires a careful consideration of the assumptions made in quantizing and sampling signals for digital processing. Also to be considered are the various alternatives available in implementation of each stage of processing. Each alternative is considered here and, when possible, analyzed to determine the relative efficiency of the technique. The basic problems to be considered are the representation of a continuous signal by a sequence of samples in space or time, the quantization of a continuous quantity by conversion to a finite precision digital number, and the limitations of the discrete Fourier transform in representing the frequency domain characteristics of a function. We are assuming here that for the purpose of treating the data analytically, the original process may be assumed to be mathematically continuous. It is possible however that the phenomenon is not truly continuous, but is experimentally sampled at a larger interval than that characteristic of the process.

The sampling theorem states that a band-limited function may be sampled at discrete intervals in the independent variable without loss of information if the interval is no larger than the Nyquist limit for the function. A band limited function has a Fourier transform which is non-zero over only a finite portion of the domain of the transform variable s. The Nyquist limit $1/2s_c$ is inversely proportional to the cutoff frequency, s_c, of the function. The synthetic data generated from the model of Chapter 1 inherently satisfies this requirement since the transform was specified to be band-limited and the space domain function computed from the transform.

Quantization of a continuous quantity by digital representation has already been considered in the discussion of quantization noise in chapter one. It was shown that the SNR for a resolution of 10^{-3} is 92 dB. This SNR is already relatively high, and all calculations are here performed with a floating point representation having a 28 bit mantissa. The SNR associated with this resolution is 158 dB within the dynamic range allowed without a change of exponent (28 bit fixed-point SNR). Due to the possible loss of precision involved in differencing numbers of similar magnitude, the accuracy of results is difficult to predict. The relative error resulting from the representation of a number x by the approximation $x'=x(1+e)$, where $e=(x'-x)/x$ is the relative error of approximation, may be greatly increased by summing two numbers of similar magnitude but

opposite sign [Knuth (1969)]. Since it is not convenient to predict loss of precision resulting from a general calculation, an attempt will be made to estimate the number of significant digits in the results given certain characteristics of the input data.

In order to estimate the precision with which the algorithm operates, some known characteristics of the input data will be used to take into account the relation between the dynamic range of the input and the precision of the algorithm output. The dynamic range of the function g(x,y) is roughly 6 orders of magnitude or 20 significant bits as presented to the processing algorithm. If it can be shown that the floating point computations preserve the 20 high order bits of the mantissa, then the computation may be assumed exact within the precision of this set of input. Since the FFT algorithm is the most used portion of the actual program, we will examine its accuracy first. The FFT computes the discrete Fourier transform of the input by a process of multiplication with phase factors W_N^{kn} and summations of these products. The phase factors all have a magnitude of 1, therefore the multiplications involved in the FFT never tend to increase the magnitude of the result, and roundoff error is significant at each stage only near the least significant bit carried in the calculation. This being the case, the troublesome factor of error magnification due to floating point multiplications changing exponents and thus perhaps incorporating erroneous data bits

into the significant portion of the next sum may be ignored here. Making the assumption that each phase multiplication does not increase the relative magnitude of the current error, we may then say that the upper limit on calculation error will be on the order of magnitude of the product of the number of successive summations and half the quantization width used in calculation. Since the quantization width used here is 2^{-28}, there may be on the order of 256 successive summations before cumulative roundoff error could begin to reduce the significance of results. For the case of white noise input to the floating point FFT, it may be shown that the output noise to signal ratio is twice the number of summations multiplied by the variance of the roundoff error in each operation [Oppenheim and Schafer (1975)]. The SNR for this implementation with a 64x64 input array is then 169 dB for white input. The floating point FFT may then be trusted to provide insignificant addition of noise to the calculations.

The same assumption may not be made regarding the portion of the algorithm which computes F_p. In this case, a floating point division is done with the magnitude of the denominator being perhaps as small as 10^{-6}, this being the limit set by the use of a tolerance factor in the computation of 1/H. There exists the possibility here of moving erroneous data bits of the numerator 20 places to the left, thereby seriously affecting the significance of the result. All other portions of the algorithm use a

normalized function (magnitude < 1) as a multiplier and therefore are not as likely to affect seriously the accuracy of the result. It is fortunately the case that the portions of F_p which are most likely to contain calculation error are also the ones which are given the least weight by the restoration algorithm.

The last basic theoretical point to be considered is that of the representation of a finite extent sampled signal by its discrete Fourier transform. In order to make use of the efficient algorithm available for computation of the FFT, the program is written to operate on an image in the transform domain whenever possible. The convolution theorem states that the transform of the convolution of two functions is the product of the transforms of the two functions. This holds for the continuous infinite representation of a function and its transform. Due to the fact that we are using discrete finite representations in both domains, several possible problems must be considered in the implementation of convolution as a product in the transform domain [Bracewell (1978)].

Implicit in the discrete representation of a function as a set of samples taken at intervals $1/2s_c$, is the replication of the transform at intervals of width s_c in the transform domain. By the convolution theorem, sampling in the function domain is equivalent to convolution in the transform domain of the transforms of the sampling function

and the original function. Since the transform of shah($x/2s_c$) is $2s_c$shah($2s_c s$), the transform of a sampled function is the superposition of many copies of the transform of the continuous function replicated at intervals of $2s_c$. This superposition of replicated copies will only be equal to the continuous function's transform if the function is band-limited with cutoff at or below s_c [Bracewell (1978)]. The phenomenon resulting when this is not the case is referred to as aliasing, since the superposition results in the addition of high frequency components from one window to the low frequency components of the next, hence aliasing high frequencies as low ones. In the function domain this corresponds to the fact that the sampled representation of a signal is not unique for frequencies beyond the Nyquist limit. The only way to avoid aliasing when sampling data of unknown spectral composition is physically to filter the data before sampling. This filtering is performed by the aperture stop in an optical system and the electronic circuitry in electrical signal detection apparatus.

Due to the finite nature (in space or time) of the sequence of samples representing the function, there is also a sampling effect on the transform. The transform of any finite continuous or discretely sampled function will be defined only at intervals of 1/2T of the frequency variable. This is a manifestation of the fact that the function is not completely specified in space. If one assumes that the

function is zero everywhere in space beyond the sampled region, then the missing values between defined transform values may be computed by interpolation based on sinc(s), or by the equivalent operation in the function domain, that of appending zero values to the sequence before performing the discrete Fourier transformation.

A result of the implicit replication in one domain of a function represented at discrete intervals in the other domain, is that the convolution represented in the transform domain as G(u,v)H(u,v) is a periodic convolution in the space domain. We refer to the convolution of two sampled finite extent functions g(x,y) and h(x,y) as a linear convolution when the operation is performed over an infinite extent window by assuming g(x,y) and h(x,y) to be zero beyond the region in which they are known (i.e., not replicated). A periodic convolution results if the assumption is made that the functions are replicated by infinite repetition of the set of sample points. The linear and periodic convolutions will give the same result only when the sum of the number of samples in g and h (in each dimension) is less than or equal to N+1, where N is the number of samples used in the transform domain representation of the result. This is equivalent to appending enough zero sample values to g or h so that there will never be simultaneous overlap of the two non-zero ends of g and h during computation of the periodic convolution.

Actual implementation of the convergent deconvolution iteration begins by computing H(u,v) from h(x,y). The experimenter must insure that aliasing does not occur in the process of sampling to produce h. The number N, the size of the FFT to be used (here assumed square, in two dimensions), must be chosen large enough such tnat $n_g+n_h-1<N$. This insures that the periodic convolutions implemented in the program will give the intended results.

$H_m(u,v)$ is then calculated as

$$H_m(u,v)=|H(u,v)|/\{\text{area}[h(x,y)]\} \tag{3.1}$$

ensuring that, for h(x,y) non-negative, $|H_m(u,v)|<1$ and therefore that the iteration will converge. Eq.(3.1) makes use of the fact that if the PSF is a real, non-negative function it has a hermitian transform with peak magnitude at the origin and equal to the area of h(x,y) [Ioup (1968), Bracewell (1978)]. This property is not, however, a requirement for the convergence of the technique (see Eq.2.8).

Next, h(x,y) is normalized in order to maintain the relationship:

$$\text{area}[f(x,y)]=\text{area}[g(x,y)] \tag{3.2}$$

realizing that in sensing the data our optical system performed the following modification of the area under f(x,y):

$$\text{area}[g(x,y)] = \{\text{area}[f(x,y)]\}\{\text{area}[h(x,y)]\}. \quad (3.3)$$

This normalization is necessary when applying constraints since we will compare F_p to G within the iteration. Any required change in area can be implemented at the end of the iterations.

Next the algorithm makes an attempt to minimize the effects of noise in g(x,y) by removing incompatible noise and attenuating compatible noise. A transform domain window is used in the new method to perform the noise removal operation. This windowing operation is

$$G_s(u,v) = G(u,v)\{1-[1-H_m(u,v)]^n\} \quad (3.4)$$

where the new, smoothed g(x,y) is computed in the transform domain by weighting the transform G(u,v) according to $[1-(1-H_m(u,v))^n]$. This weighting is equivalent to performing n iterations of Morrison smoothing in the time domain with the function $h_m(x,y)$ in place of h(x,y) [Morrison (1963), Ioup (1968)].

The new technique now makes an approximation of f(x,y) as $f_p(x,y)$ defined above, using a tolerance factor of 10^{-6} in computing 1/H(u,v) to avoid arithmetic overflows in the computation of $F_p(u,v)$ and limit the effects of errors introduced by the calculation. The tolerance is chosen much smaller than the quantization interval used in sampling the model output and therefore functions solely to prevent overflow and limit calculation error. This tolerance factor

is applied simply by avoiding the division 1/H(u,v) whenever |H(u,v)|<tolerance and substituting zero for the result. Due to the efficiency of computation in the transform domain, all functions are represented in the program by their transforms most of the time. Only the application of function domain constraints is done in the space domain.

Application of constraints is performed by inverse transforming the current $F_i(u,v)$ and then forcing $f_i(x,y)$ to meet the constraints input to the program. The time domain function is corrected to meet the peak, non-negativity, and finite extent constraints and then written out to a disk file. Advantage is taken here of the fact that the imaginary part of $f_i(x,y)$ should be zero, and it is cleared before continuing the iteration. This should help to reduce the propagation of roundoff errors through the iterations. The iteration then proceeds by transforming the constrained $f'_i(x,y)$ back to $F'_i(u,v)$ and repeating the previous procedure until the repetition count is exhausted.

CHAPTER 4

RESULTS AND CONCLUSIONS

In order to test the usefulness of the iterative smoothing and deconvolution techniques of chapter three for image data, the algorithms are here applied to several sets of synthetic input data generated by the model of chapter one. The first set of data, used as an initial test for convergence and symmetry, is the response of the defocused model optical system to an input with 270 degrees of circular symmetry. The intensity distribution of f is a cylindrical Rect(r) with a smooth gaussian edge over 270 degrees of arc and a sharp edge over the remaining quadrant. These data are processed both with and without the addition of noise.

The second set of data used in testing the application of constraints and the effects of noise on the method is chosen to provide a qualitative measure of performance for several types of objects. The first is a pair of delta functions separated by approximately one half of the half-width of h(x,y). The second is a pair of narrow gaussian peaks separated by the same distance. The third is

a sharp right angle shaped object with arms fifteen samples long and two samples wide. The fourth object is a right triangular wedge with discontinuous sides. Each object has unit peak intensity. Due to the lower response amplitudes generated by the model for sharp, narrow objects, the restoration of the deltas, Gaussians, and angle will require raising the height of their peaks significantly. The peak constraint will then be less effective on the first three objects than on the large triangular object.

The third and fourth sets of data used in testing are two different noisy cases of the second set. Approximately white gaussian noise is added to both sets in different amounts. The SNR for set three is 200:1 or 23 dB and for set four the SNR is 2000:1 or 33 dB. The high noise level corresponds to a relatively noisy image and the lower noise level represents a fairly clean image [Andrews and Hunt (1977)].

Presented in chapter one were the displays of the model PSF and OTF. The same PSF and OTF are used for all of the following data. These are the defocused PSF and OTF from chapter one. Following is the set of plots representing the first set of test data. Figure 4.1 is a display of f(x,y) as input to the model to generate g(x,y). The sharp edge and flat top will test the performance of the restoration method. Figure 4.2 shows the Fourier transform of f(x,y). Note the ripples occurring parallel to the sharp edge (of

f(x,y)) and extending out to high spatial frequencies. The output of the system, g(x,y) is shown in Figure 4.3. Just evident in the response is a flattening of one side of the peak where the sharp edge of f(x,y) occurred. Also note that the flat top has been completely rounded off. Evident in G(u,v), Figure 4.4, is the attenuation of the high frequency components of F(u,v) and the introduction of a negative oscillation resulting from the multiplicative negative lobe of the blur OTF in the model. Figure 4.5 is a display of $f_p(x,y)$ computed for this g(x,y) and the model PSF. Since there was no significant noise in g(x,y), $f_p(x,y)$ is very close to f(x,y) although the effects of calculation noise are apparent in the flat areas. Figures 4.6 through 4.7 present the results of performing 1,5, and 10 iterations of deconvolution on g(x,y). It is evident in the first iteration that the technique is beginning to restore the sharp edge to the data and to flatten the rounded peak. The fifth and tenth iterations indicate that the iteration is rapidly converging to f(x,y). The non-negativity constraint has been applied to these data at the end of each iteration.

In order to demonstrate the effect of noise on the results from this technique, the same run is repeated after adding white gaussian noise to achieve a signal to noise ratio of 700:1. As is evident from Figure 4.9, this amount of noise is not readily visible on the data, yet the effect on the principal solution is to render it useless as a final

result (Figure 4.10). Even so, Figures 4.11 through 4.13 indicate that useful results may be obtained from the iterative technique for small numbers of iterations even without first performing noise removal.

The second set of test data contains four separate objects to provide a measure of performance for various types of input. Figure 4.14 is a display of this f(x,y). Two of the objects are pairs of small, sharp objects in order to test the resolution of the result. All objects have the same peak intensity, but the total power radiated by each object varies considerably between the pair of point sources and the wedge. The output of the model system, g(x,y), displayed in Figure 4.15, is greatest for the wedge and smallest for the pair of point sources. A plot of log[1+g(x,y)] is shown in Figure 4.16 to indicate more clearly the smoothing effect of the model. The objective of restoring these data is to bring the amplitudes back to their original levels by putting the spread-out power back where it belongs, thereby sharpening the image also. Again, the data are first deconvolved without added noise in order to verify the accuracy of the implementation, and to observe the effects of various combinations of constraints.

Unfolding (deconvolution) iteration number ten, with all constraints applied between each iteration, is shown in Figure 4.17. Obvious improvements have been made in the sharpness of edges and the relative amplitudes of the small

objects. Figure 4.12 displays the result of applying the constraints only once, after the equivalent of ten iterations performed in one step in the transform domain. The lessened effect of the processing on edges and the flat top of the wedge is apparent, but the deltas still show about the same improvement over the input. Processing in this manner is of interest due to its possible savings in computer time. Figures 4.18 through 4.37 provide a comparison of results of processing with the application of each constraint individually and all constraints together at each iteration and at every tenth equivalent iteration for 10, 20, 50 and 100 iterations. A comparison of Figures 4.33 and 4.34 indicates that, for these noiseless data, only ten applications of constraints (once every ten equivalent iterations) has provided nearly the same result as the much more expensive application of constraints at each of the 100 equivalent iterations. Apparent from a comparison of the four sets of processed data is the fact that application of constraints individually or jointly has a cumulative effect on the result, the magnitude of which is dependent on the number of applications. Most effective for these data is the non-negativity constraint for speeding convergence and increasing bandwidth. Also important for the "bright" portions of the image is the application of the upper limit constraint. It tends to provide a similar additional advantage in regions of the image which approach maximum brightness. For this image the main effect of the finite

extent constraint is to limit the extent of Gibbs oscillations resulting from the sharp edges of the objects.

The next set of data was generated by adding gaussian noise to f(x,y) to obtain a SNR of 200:1. The noise is visible in Figure 4.38. A plot of log[1+g(x,y)] is shown in Figure 4.39 to indicate the actual SNR for each of the four objects. The actual SNR for the pair of point sources is only 1.5:1, and this noise level is therefore very likely to obscure important details necessary for complete restoration of these features. Five iterations of noise removal were first performed on g(x,y) to achieve a large degree of noise removal (and high frequency attenuation). A total of thirty unfolding iterations and constraint applications were then performed, the first 10 with an equivalent iteration interval of 2, the next 10 with an interval of 5, and the last 10 with an interval of 10. The final result, Figure 4.40, is significantly sharper than the original with little detail lost in noise. Figure 4.41 shows the result of applying only 20 iterations, the first 10 at an interval of .5 equivalent time domain iterations, and the next 10 at an interval of 4. Prior to deconvolution, 20 iterations of smoothing were applied. Since this number of smoothing iterations corresponds to significantly less high frequency attenuation of G(u,v), the deconvolution converges more quickly, though it also shows a greater amount of noise. Figure 4.42 shows the result of continuing out to 105 equivalent iterations. It is apparent from this result that

the optimum number of iterations is less than 105 since the noise amplitude has increased without a significant improvement in sharpness or amplitude of the restored point sources.

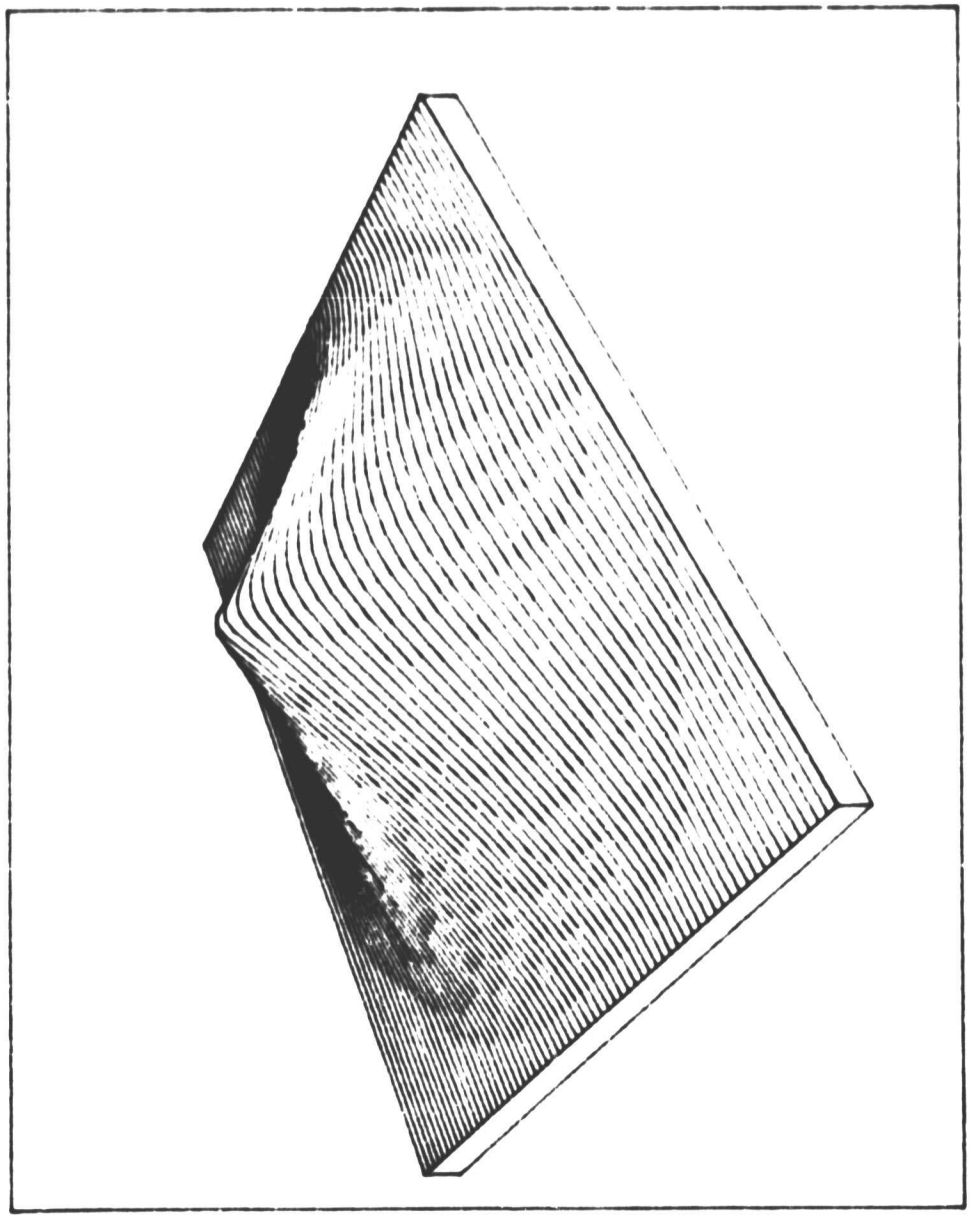
Figures 4.43 and 4.44 show the effect of not using the upper limit constraint for two different numbers of iterations of smoothing and unfolding. It is apparent from these displays that the peak constraint is of great value only for the wedge, it does not aid the restoration of the lower power objects. The remaining test runs on these data were all done after applying 20 smoothing iterations to g(x,y). These results show a definite correlation between the speed of restoration and dominance of noise and the number of times constraints are applied in computing the result. Further study could be directed to determining an optimum interval for application of constraints based on the amount of smoothing and the SNR.

The last set of data processed had a SNR of 2000:1. Again, f(x,y) consisted of the same four objects. In order to show that inverse filtering is not adequate for even this low noise level, $f_p(x,y)$ is displayed in Figure 4.50. Application of 60 iterations of noise removal, however, improves $f_p(x,y)$ considerably to that shown in Figure 4.51. Figures 4.52 and 4.53 show the results of deconvolving g(x,y) an equivalent of 100 and 300 time domain iterations with 20 and 40 constraint applications respectively. In

this set of data the point sources are being restored much more rapidly due to the fact that less attenuation of high frequencies is necessary in the smoothing operation. Symmetry of the point sources is also good at this lower actual SNR.

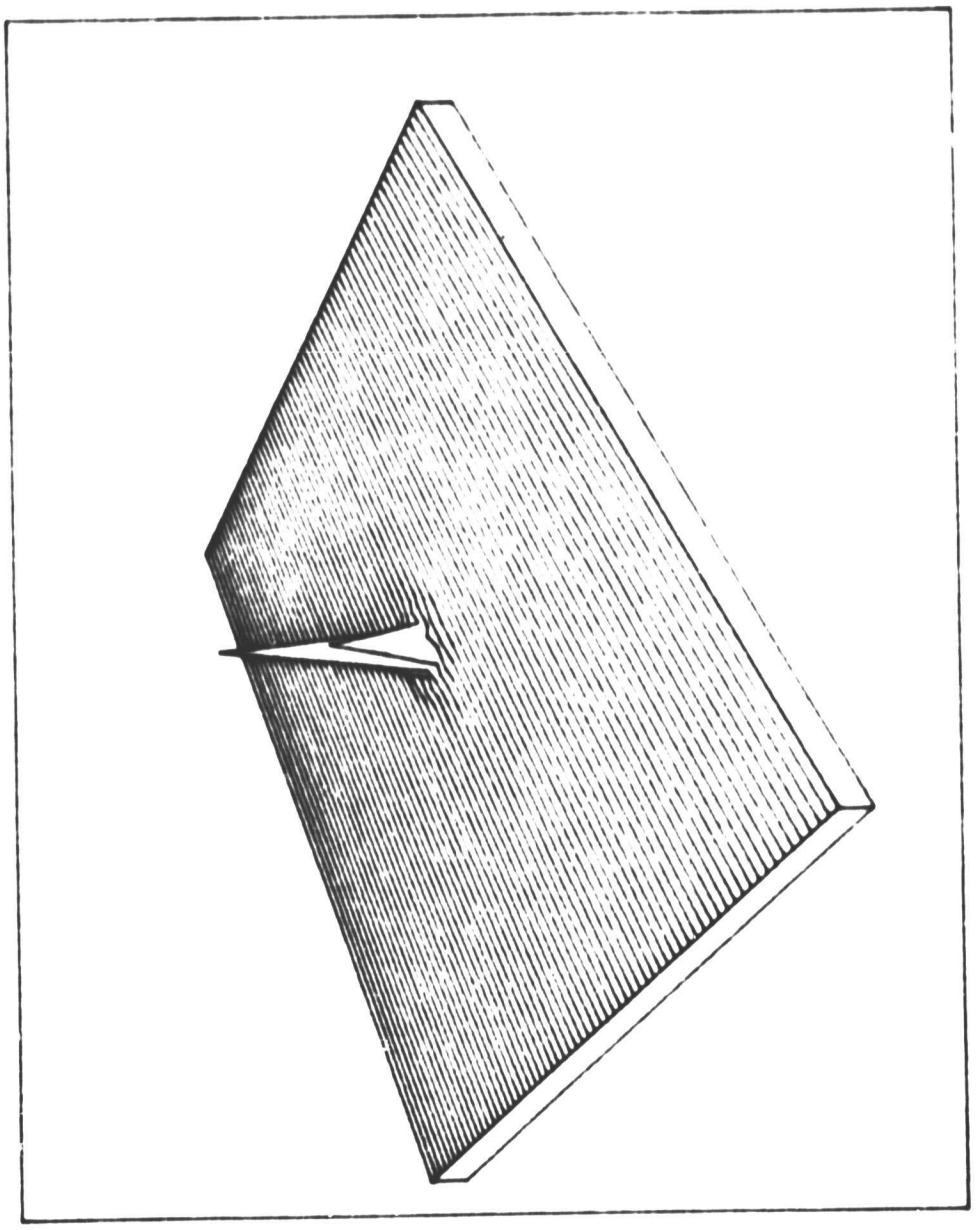
The results discussed above demonstrate the effectiveness of the convergent iterative techniques in accomplishing noise removal and deconvolution of optically sensed data suffering from severe defocusing effects. For the model studied, the PSF and OTF were accurately determined, but this is not always possible with experimental data. Since these methods do not rely heavily on accurate knowledge of these system characteristics for convergence, it is expected that useful restoration of imagery could be obtained by estimating the system PSF and OTF [Yoerger (1979)]. Ability to use this technique to apply constraints after any number of equivalent time domain iterations makes it extremely flexible for using constraints to remove noise from the image and increase its bandwidth while deconvolving. The most effective constraint for this particular f(x,y) was the non-negativity constraint, and this suggests that the method could be applied to astronomical data with great success, since such images often consist of point sources on a black field.

In addition to investigating means of determining an optimum number of iterations of deconvolution and constraint applications for the new convergent iterative deconvolution technique, further study is indicated to improve the approximation $f_p(x,y)$ used in the iteration. Since this function "pulls" the result toward an unacceptable limit in the case of noisy data, it would be desirable to include the effects of the constraints into $f_p(x,y)$ as the iteration progressed. It is apparent from the results presented here that random noise is attenuated by the application of constraints. Greater benefit might be derived from these constraints if a way is devised to incorporate their noise cancelling properties into $f_p(x,y)$ while iterating in order to allow a greater number of iterations to be performed before noise emphasis begins to dominate restoration. Further study is also indicated to investigate the effects of constraints on resultant bandwidth of processed data by studying the spectrum of each iteration. Also of interest is the effect of replacing the initial estimate $f_p(x,y)$ with other reasonable estimates for the deconvolved image such as the Wiener filter solution. It is possible that intermediate iterations using this technique to apply constraints will provide an improvement over the one-shot filtering technique when it is used in this manner.



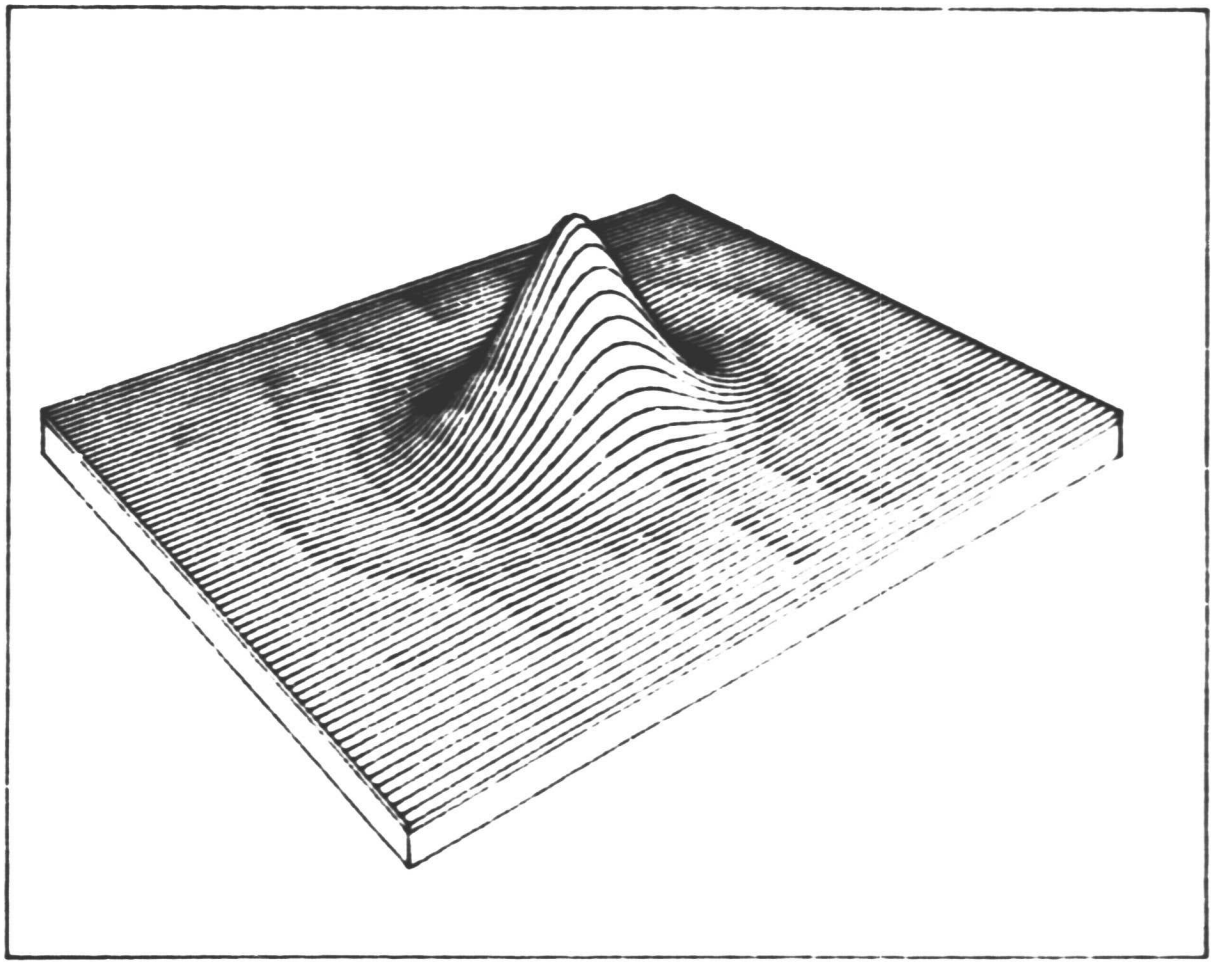
FOCUSED OTF

Figure 1.1



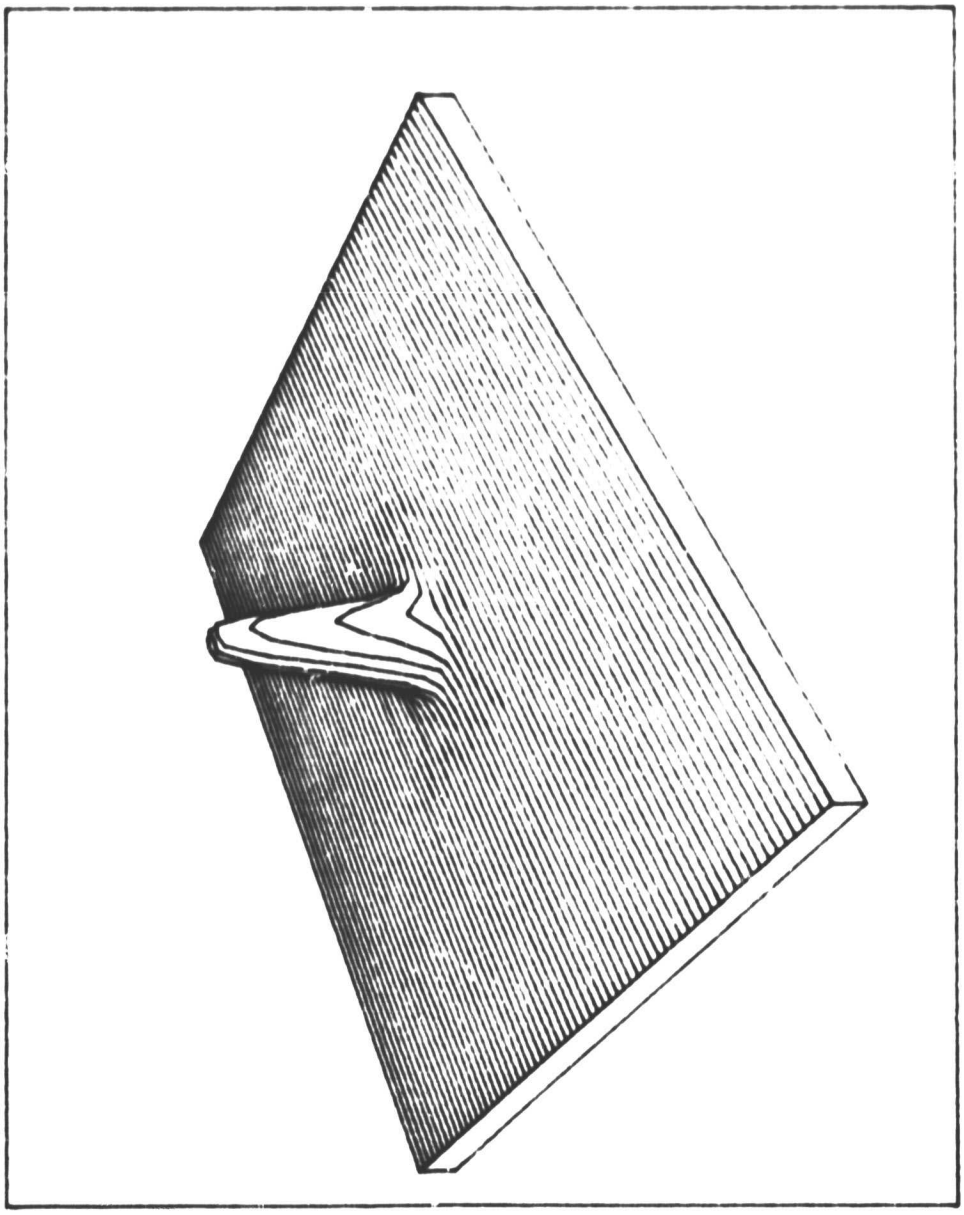
FOCUSED PSF

Figure 1.2



DEFOCUSED OTF - RUN NUMBER 1

Figure 1.3



DEFOCUSED PSF - RUN NUMBER 1

Figure 1.4

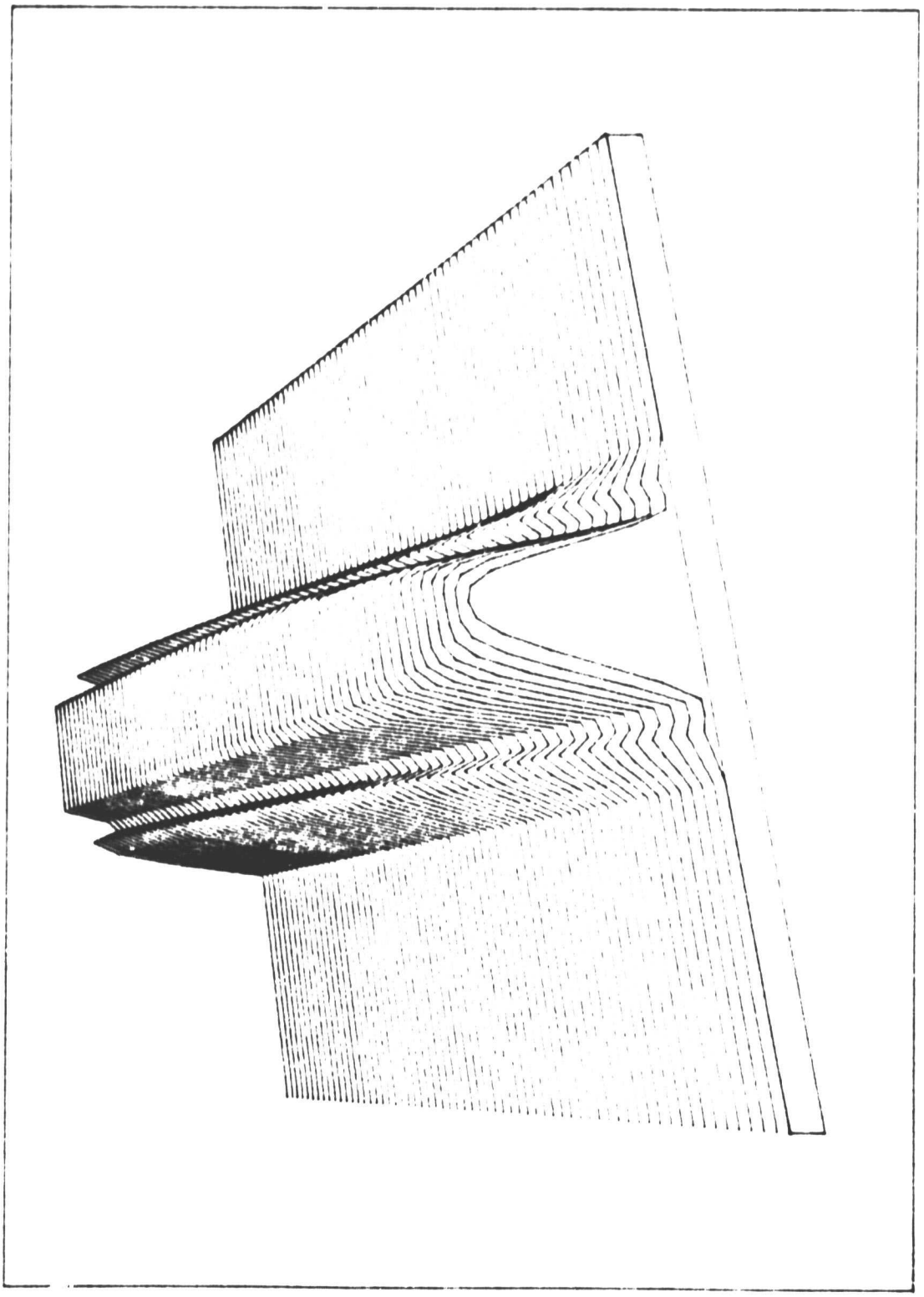
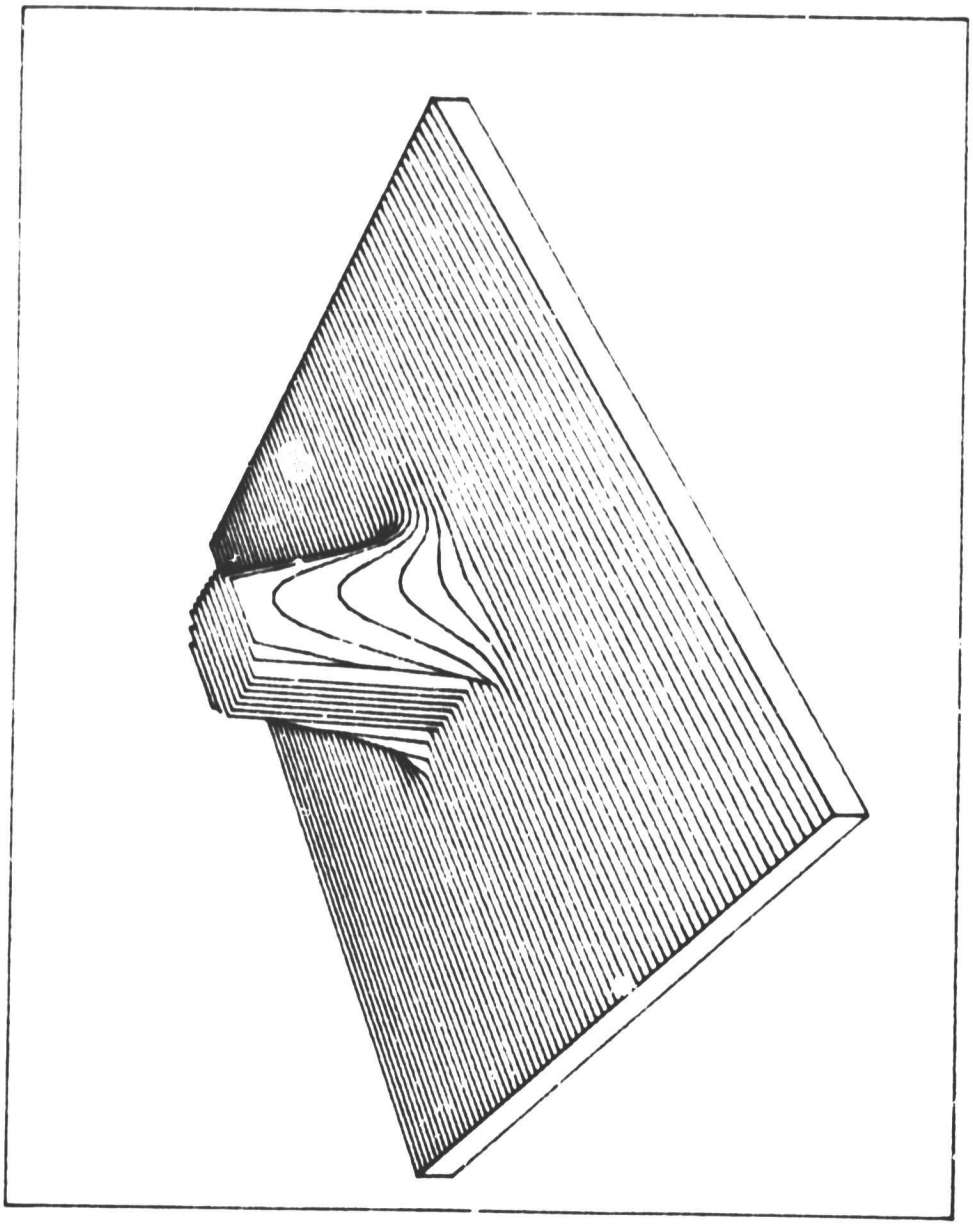


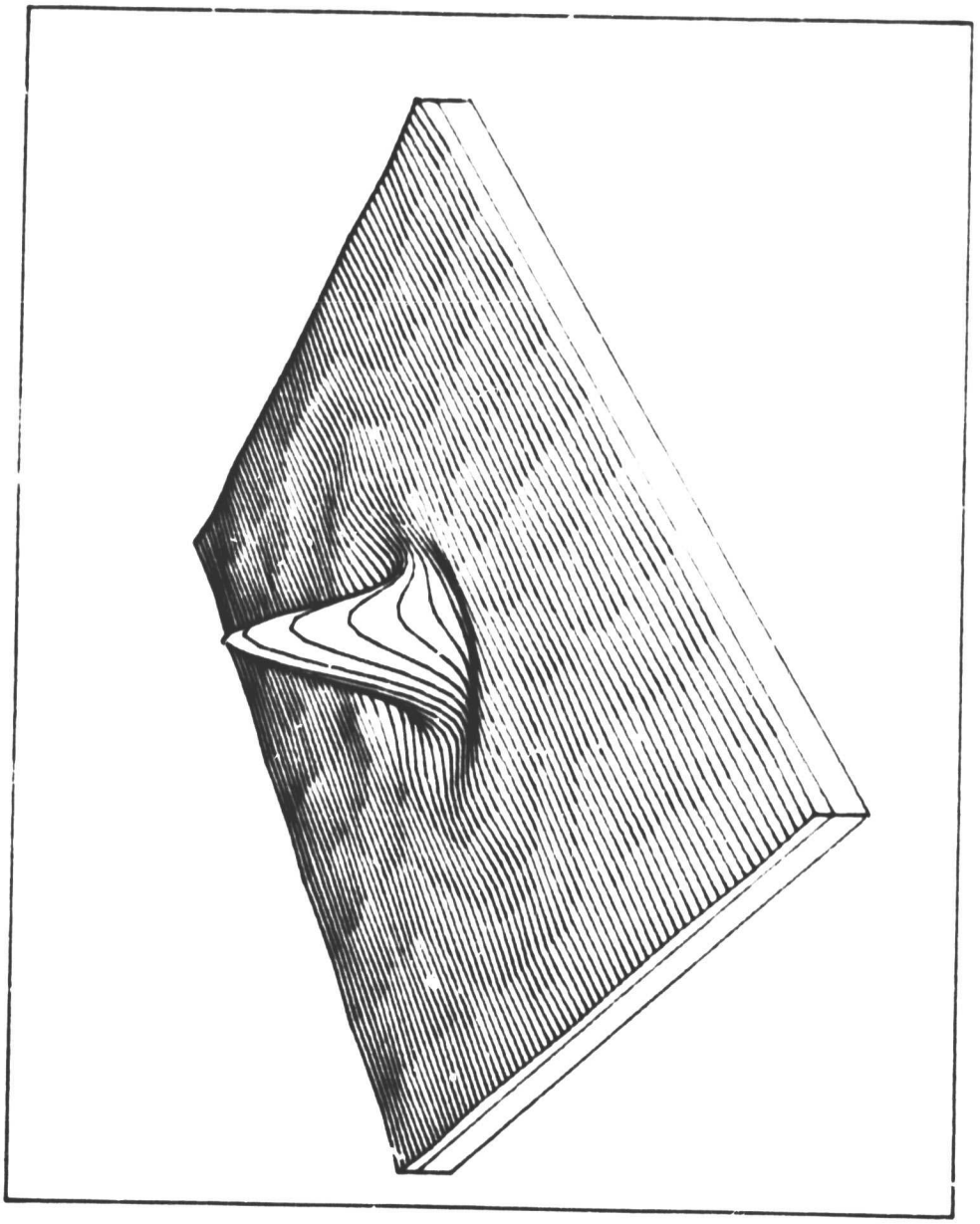
Figure 2.1

Convergence plot



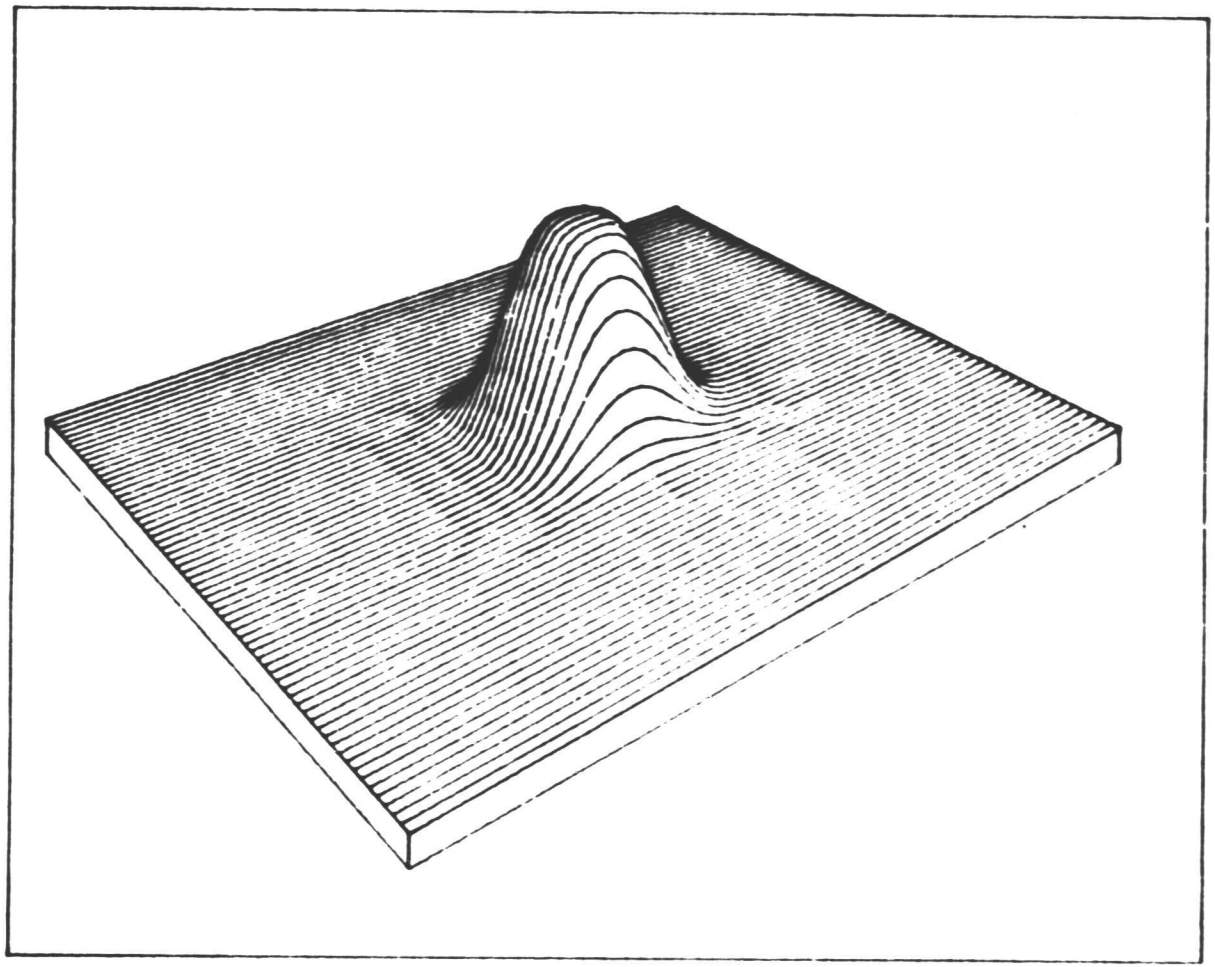
f.dat

Figure 4.1



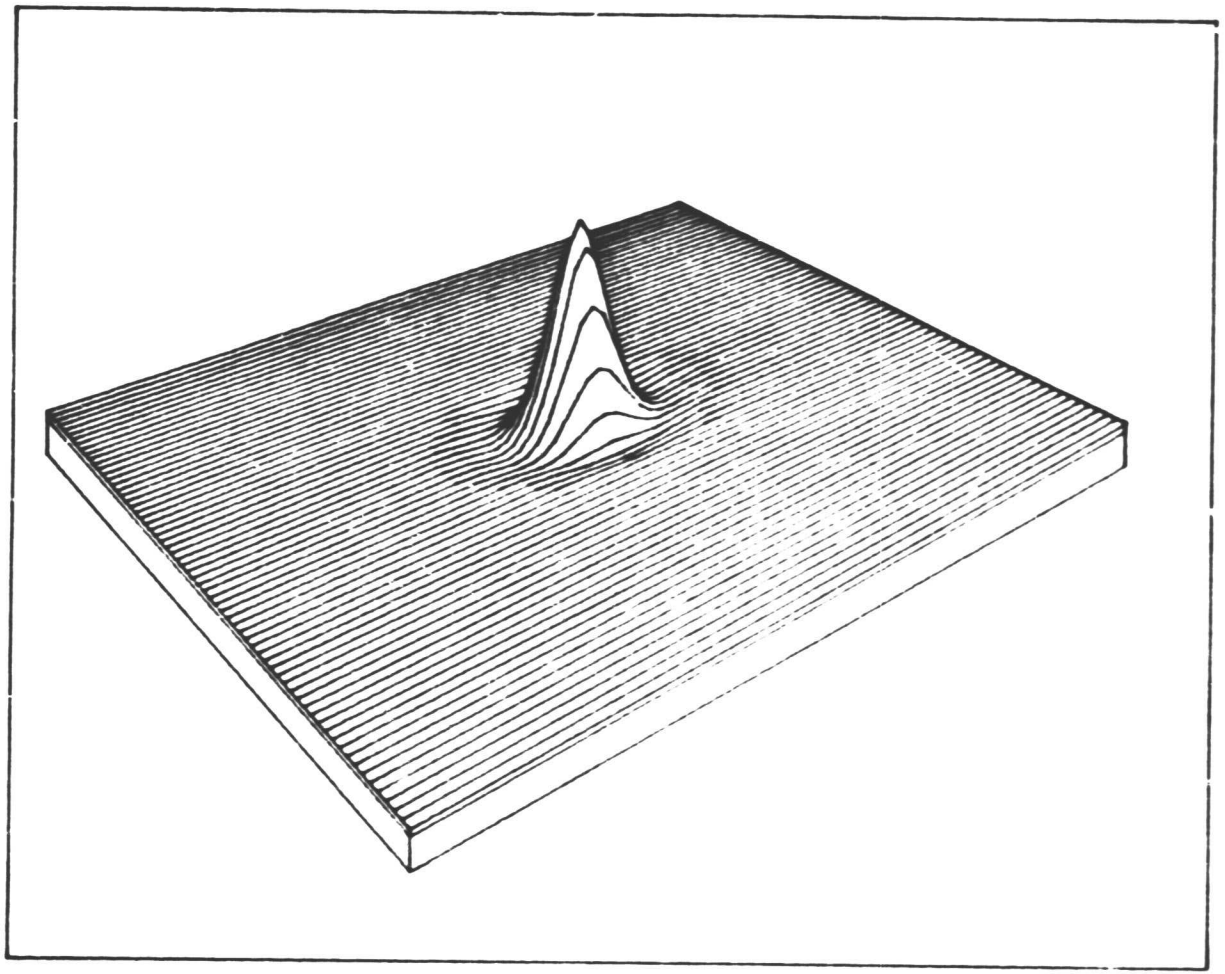
F - -1 TRANSFORM OF f

Figure 4.2



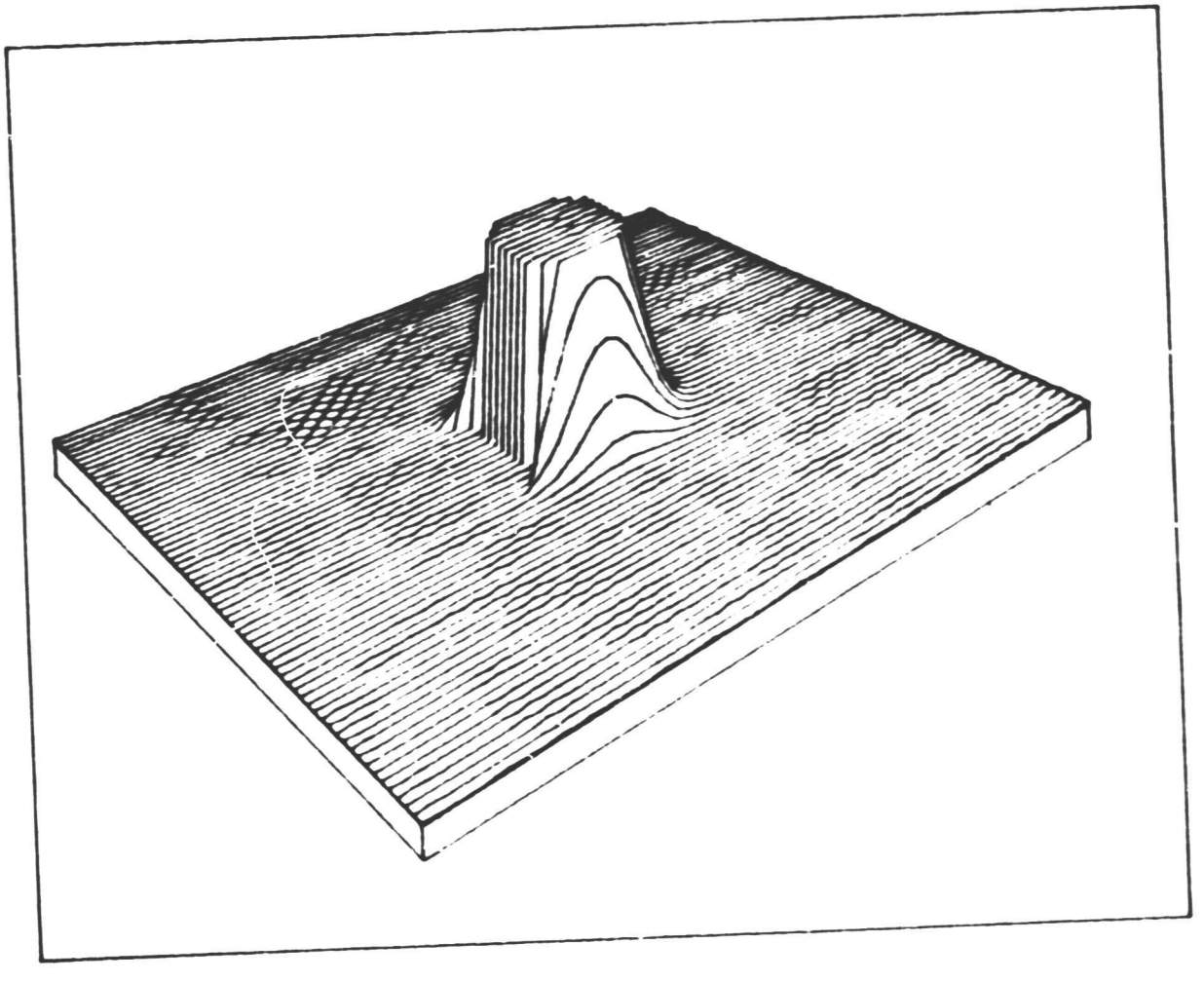
Run 1 : g(x,y)

Figure 4.3



Run 1 : G(u,v)

Figure 4.4



f_p

Figure 4.5

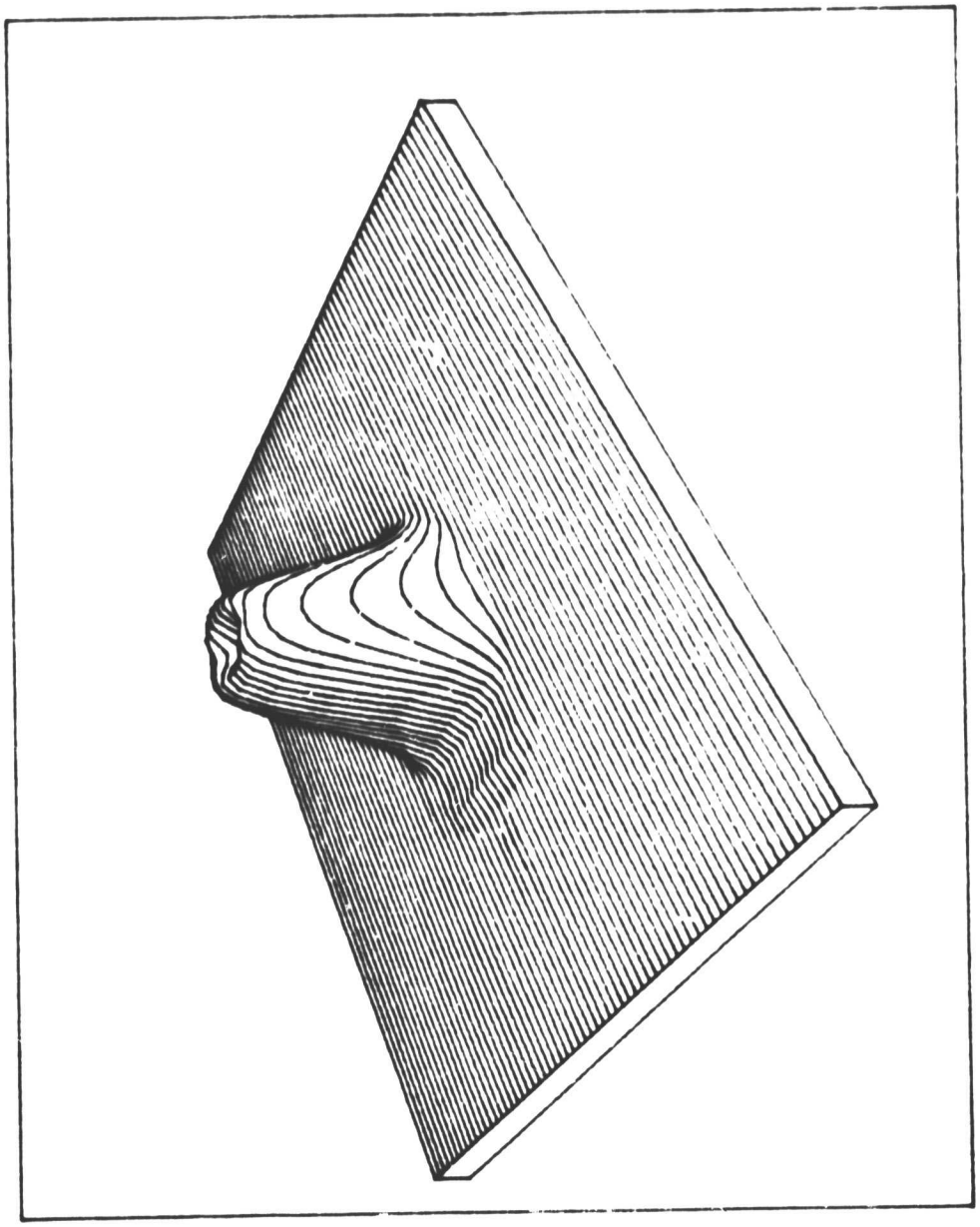
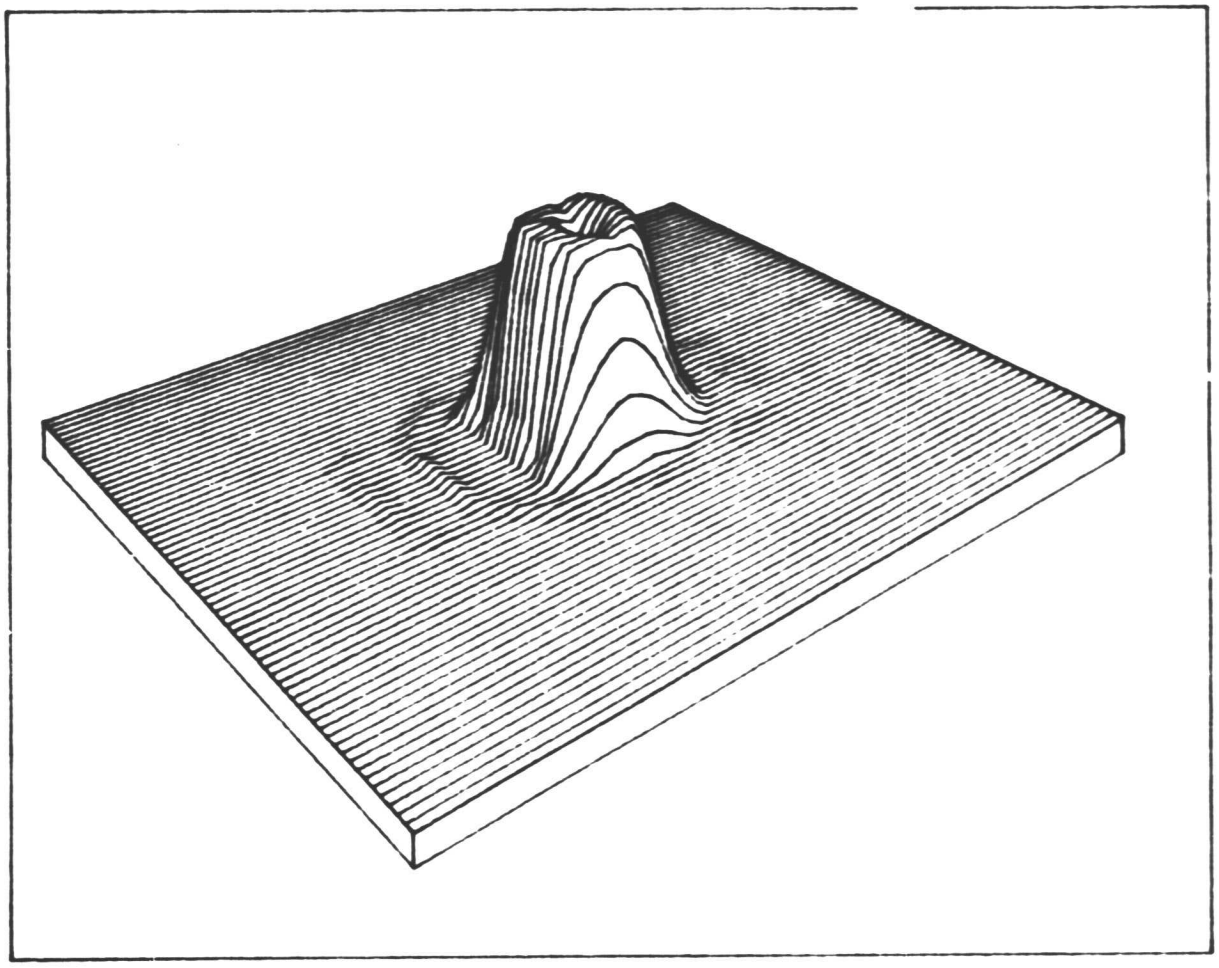
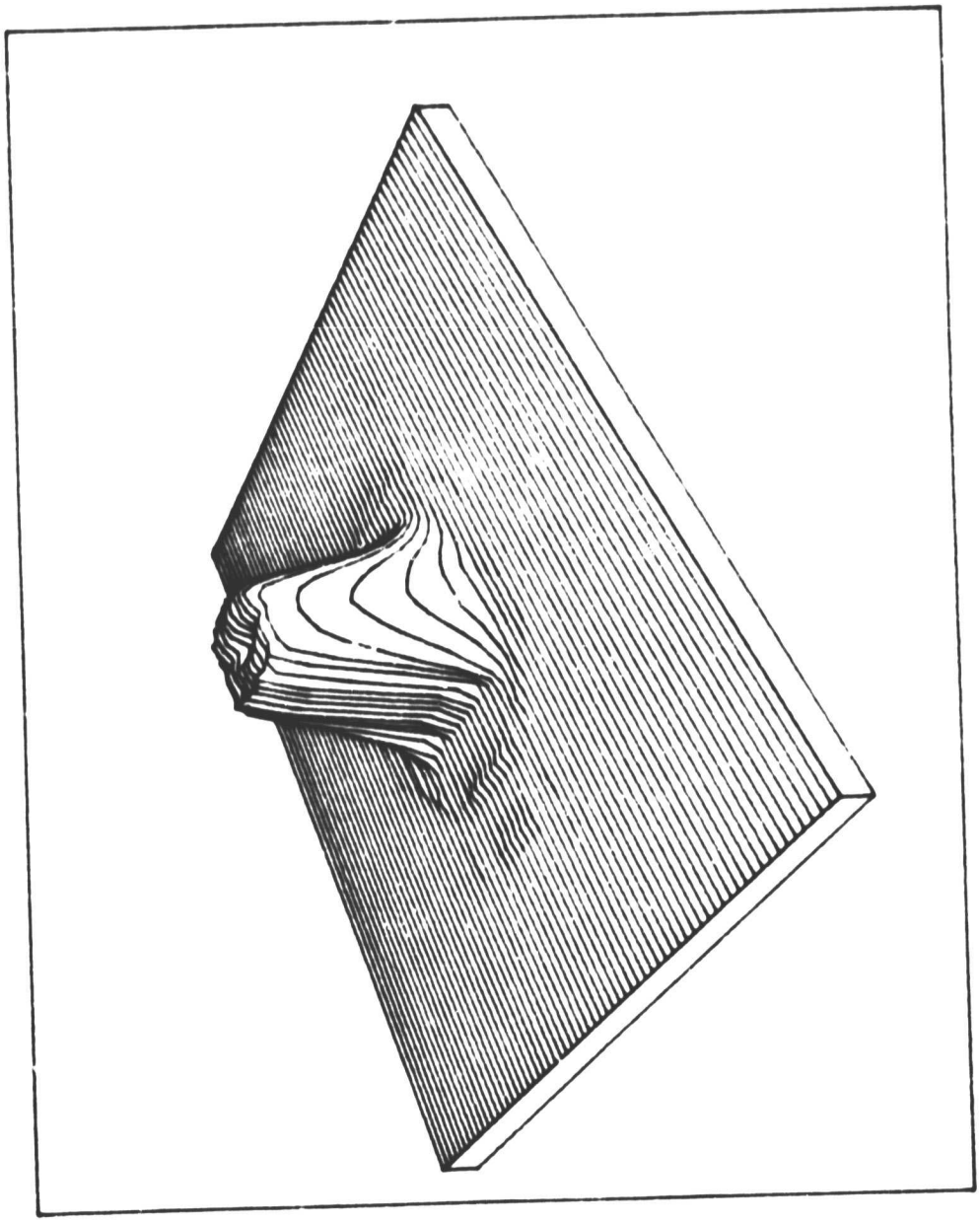


Figure 4.6



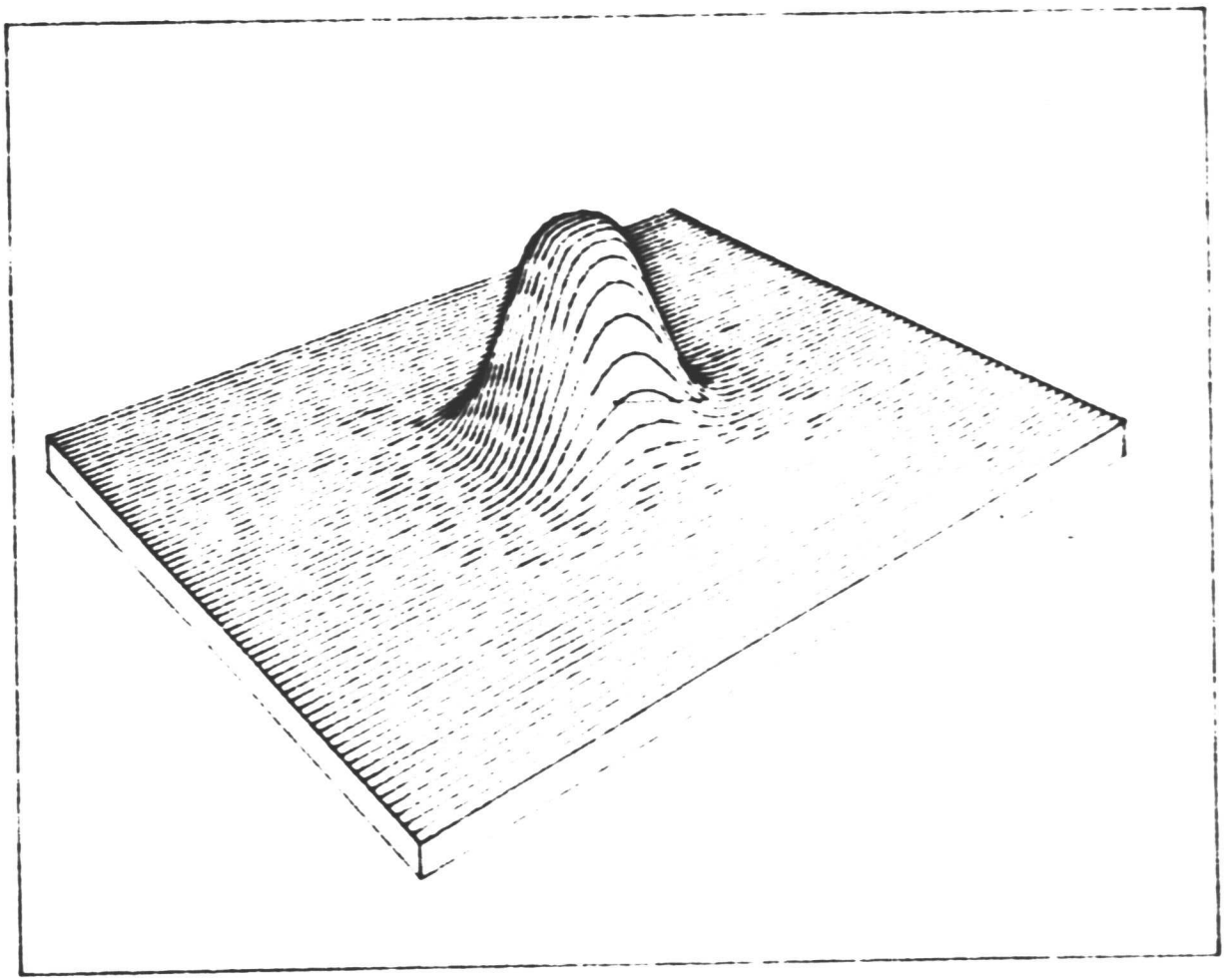
UNFOLDING ITERATION #5 - RUN NUMBER 1

Figure 4.7



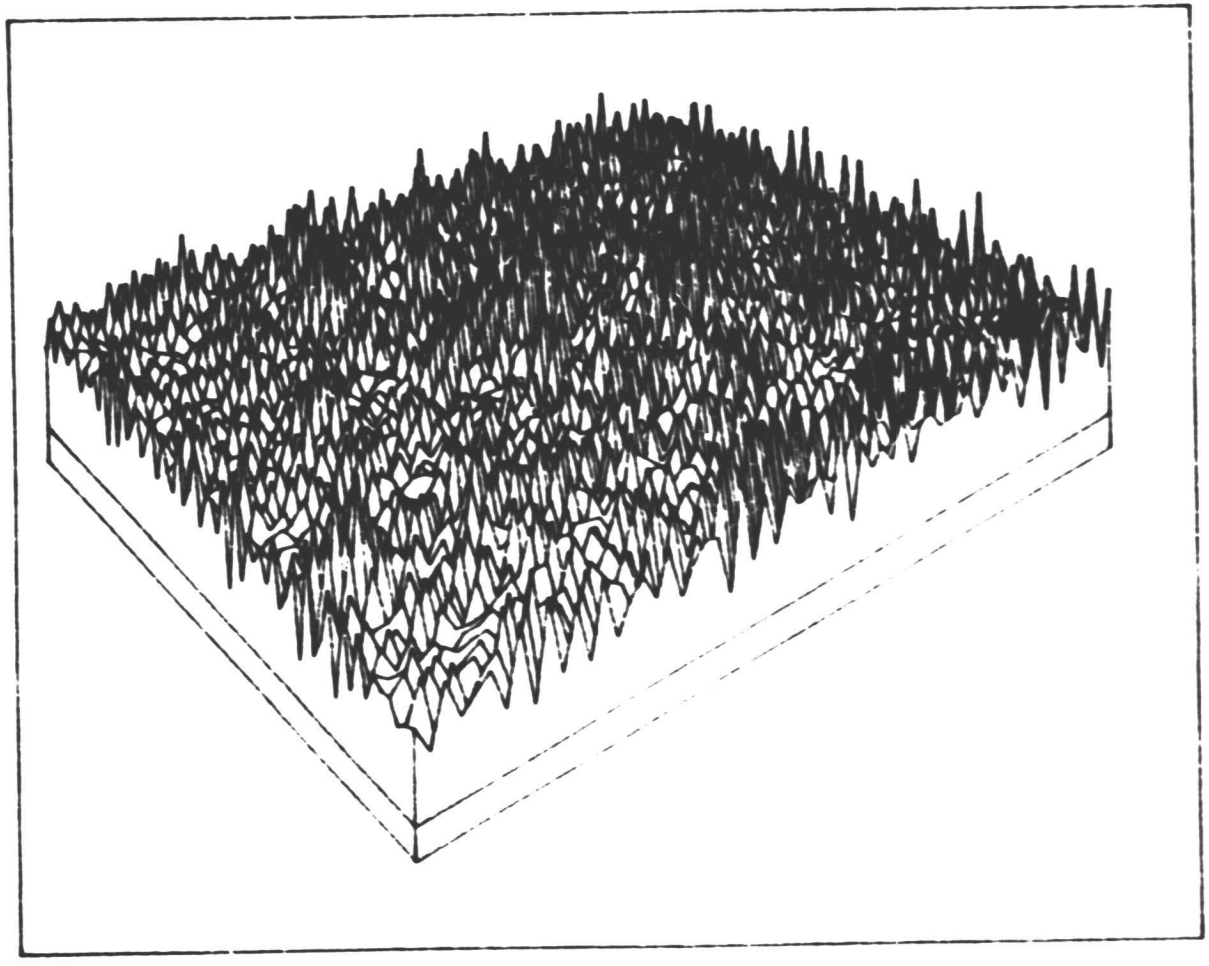
UNFOLDING ITERATION #10 - RUN NUMBER 1

Figure 4.8



g(x,y) + noise ; S/N=700

Figure 4.9



f_p NOISE RUN NUMBER 1

Figure 4.10

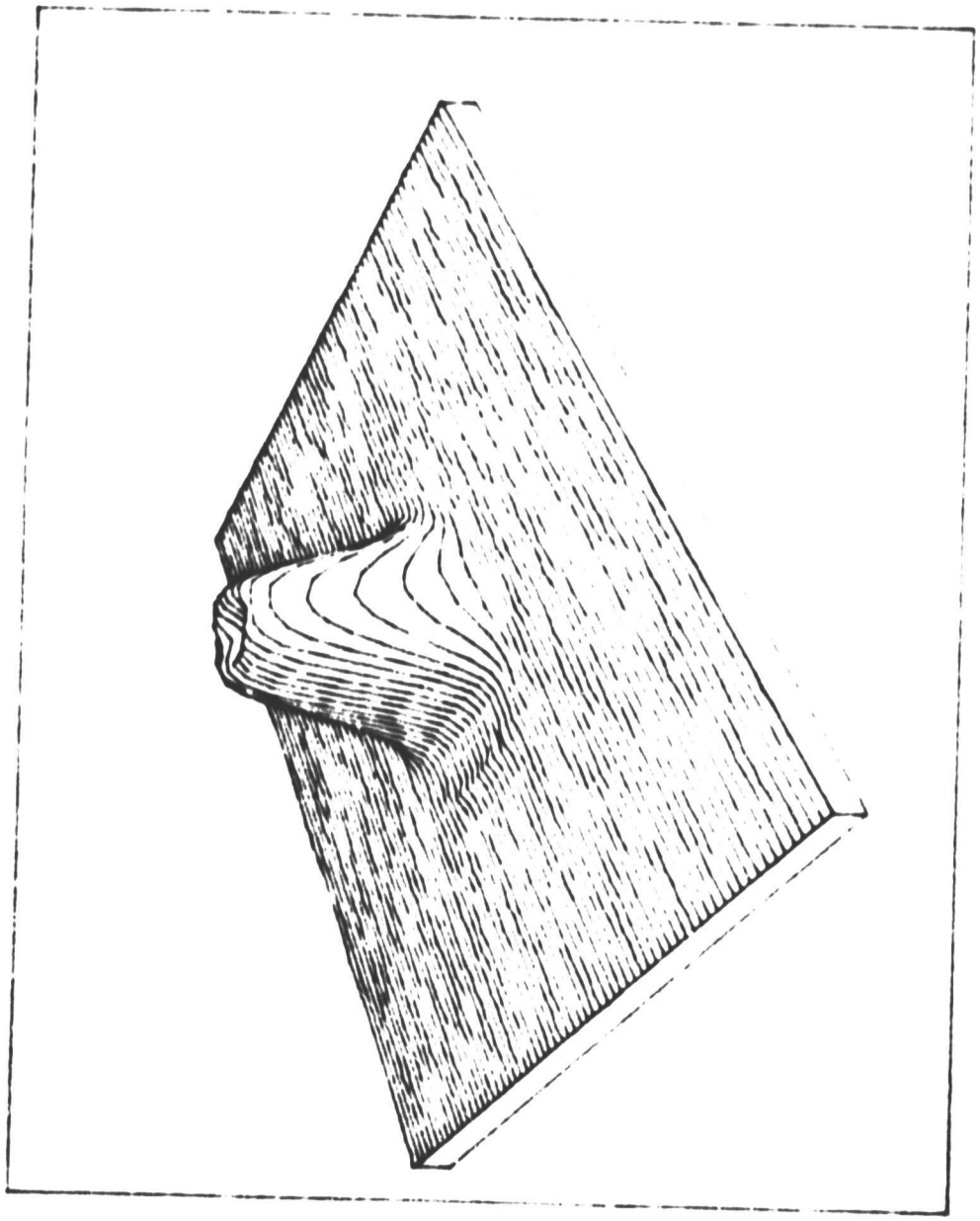
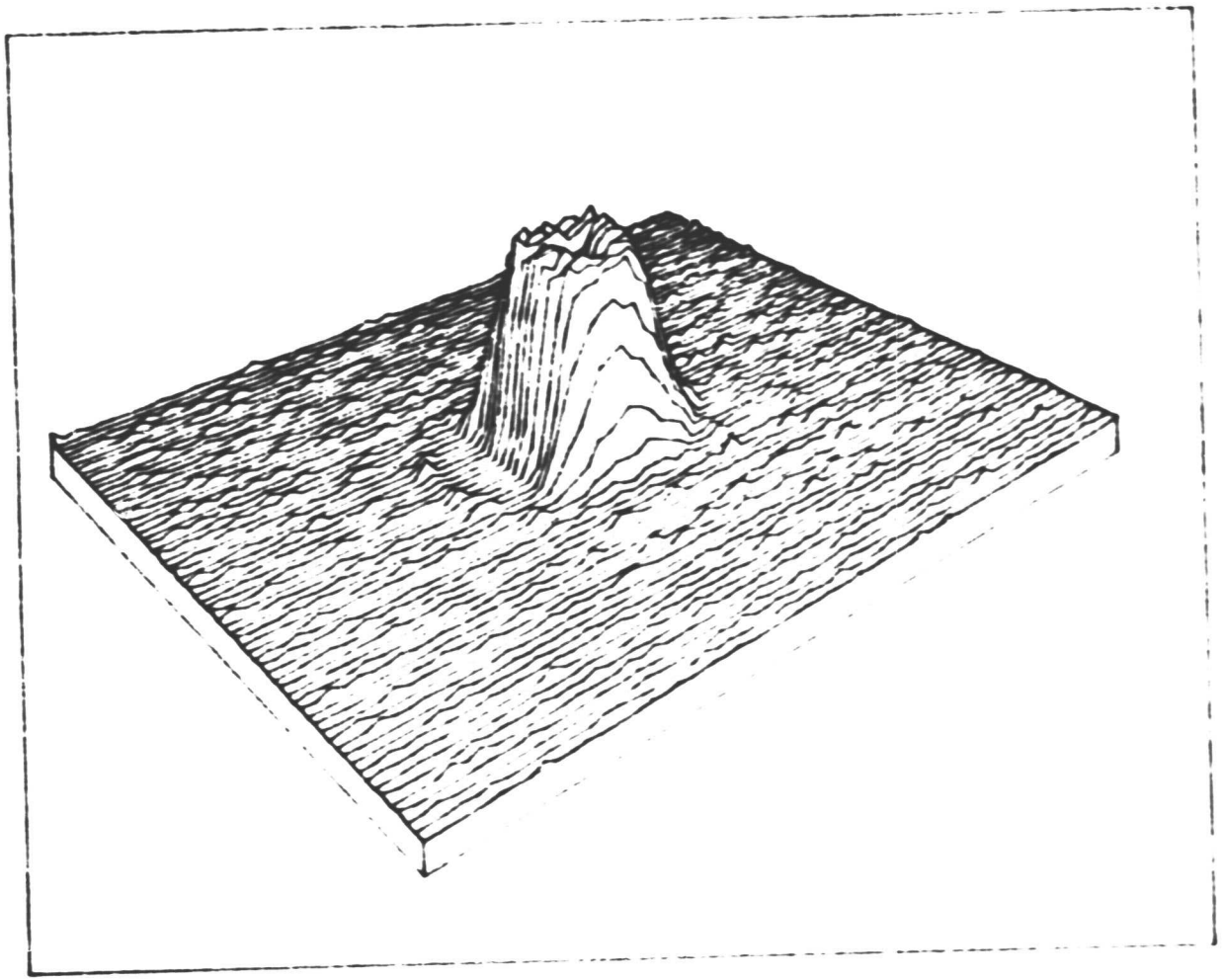
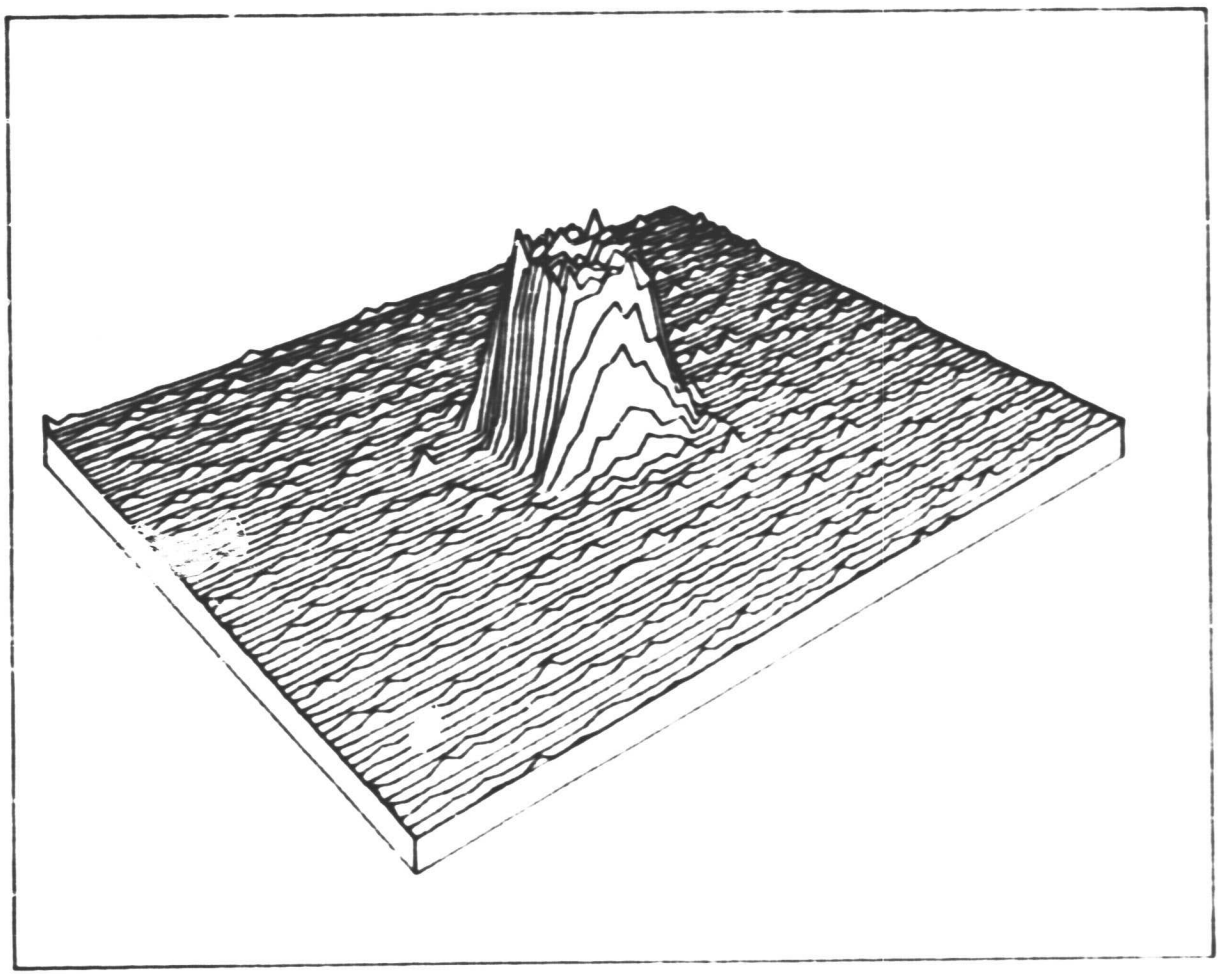


Figure 4.11



UNFOLDING ITERATION #5 - NOISE RUN #1

Figure 4.12



UNFOLDING ITERATION #10 - NOISE RUN #1

Figure 4.13

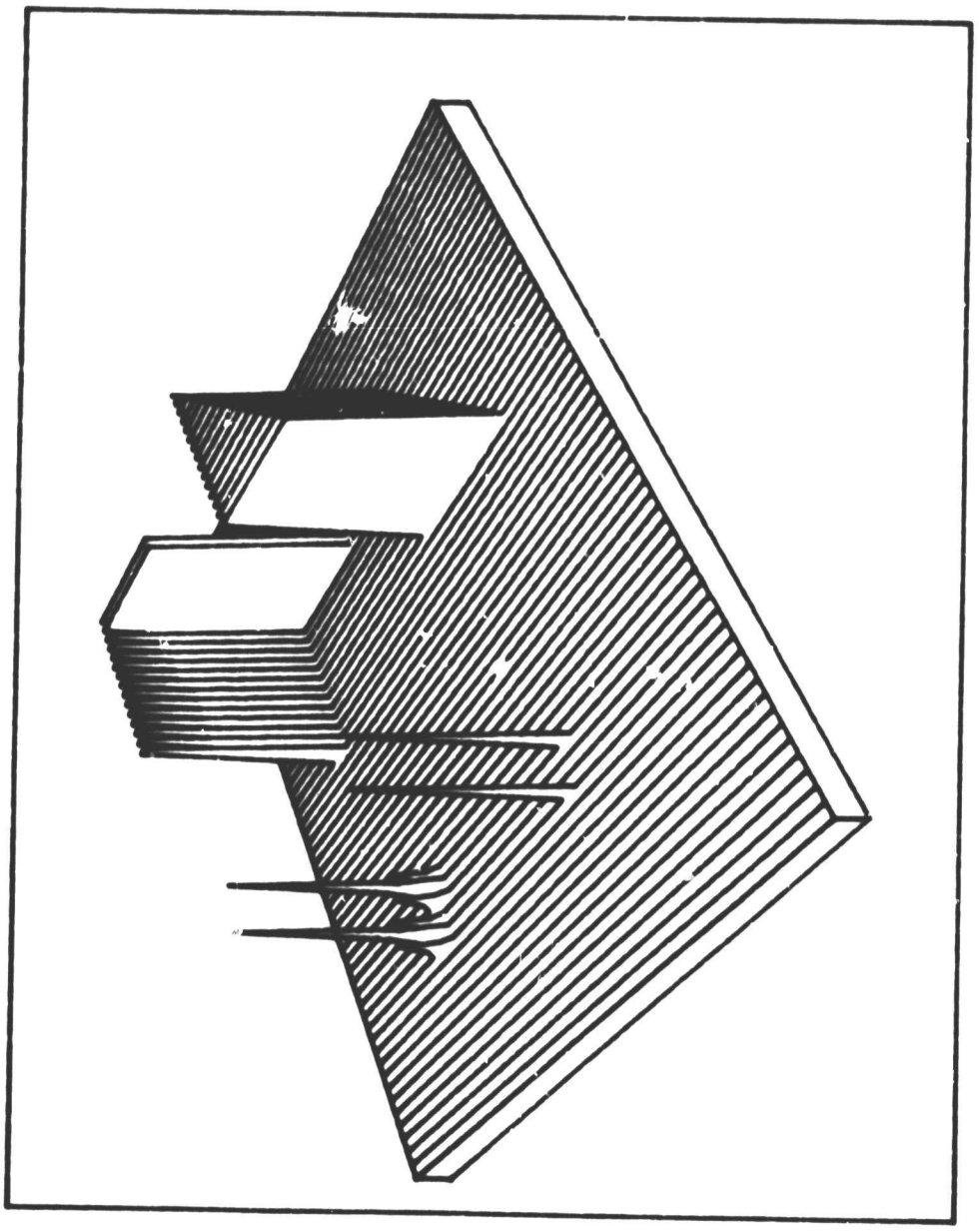
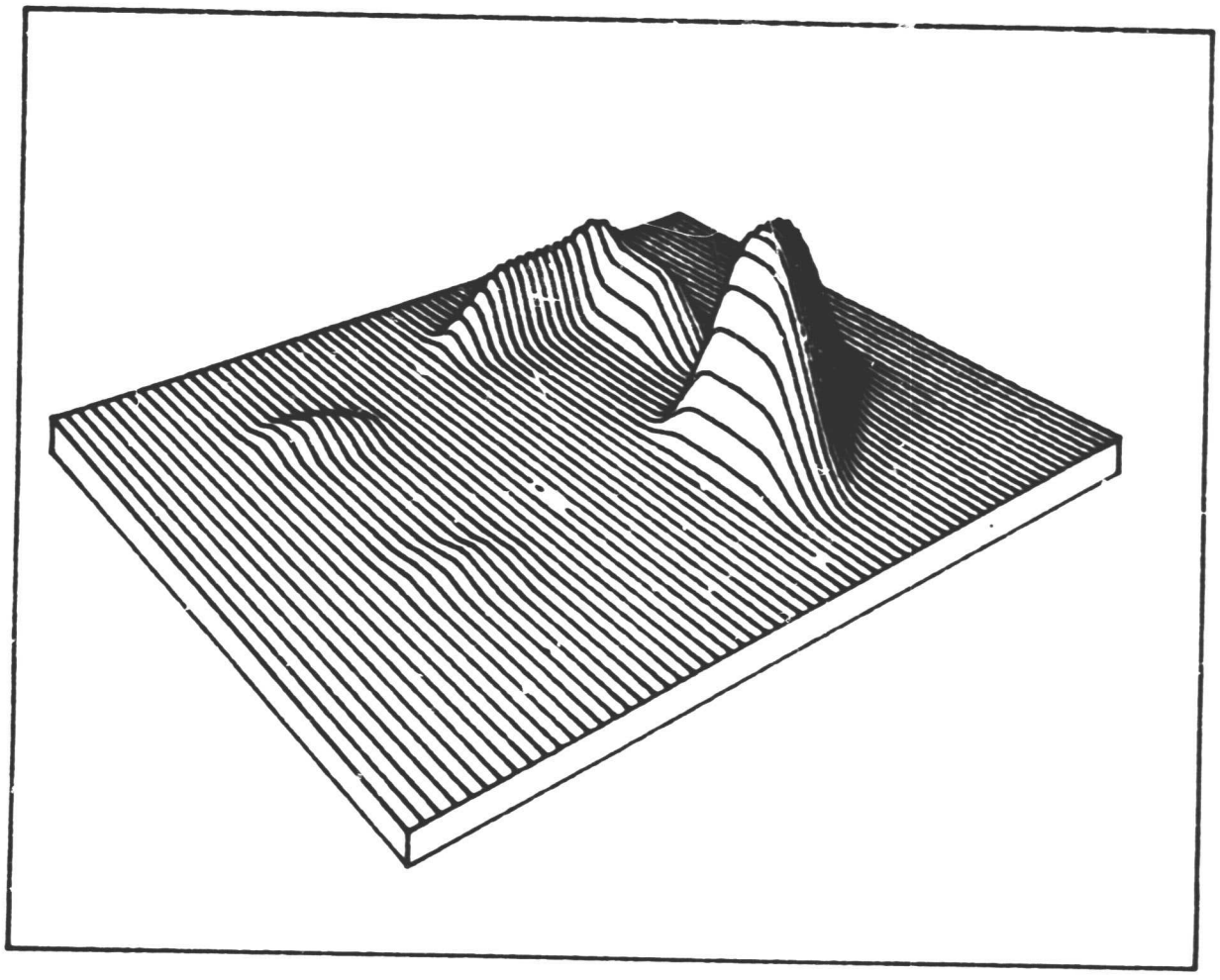
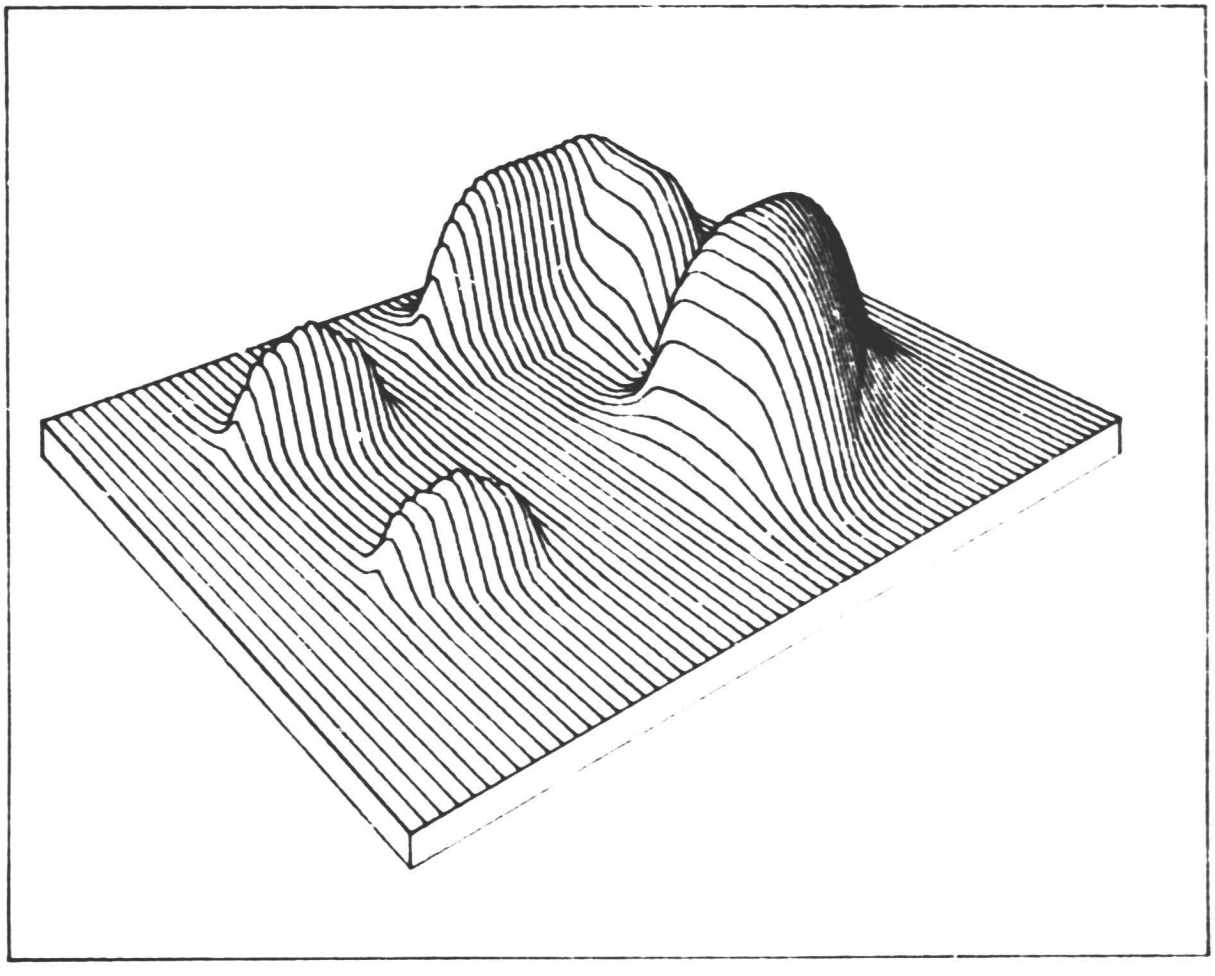


Figure 4.14



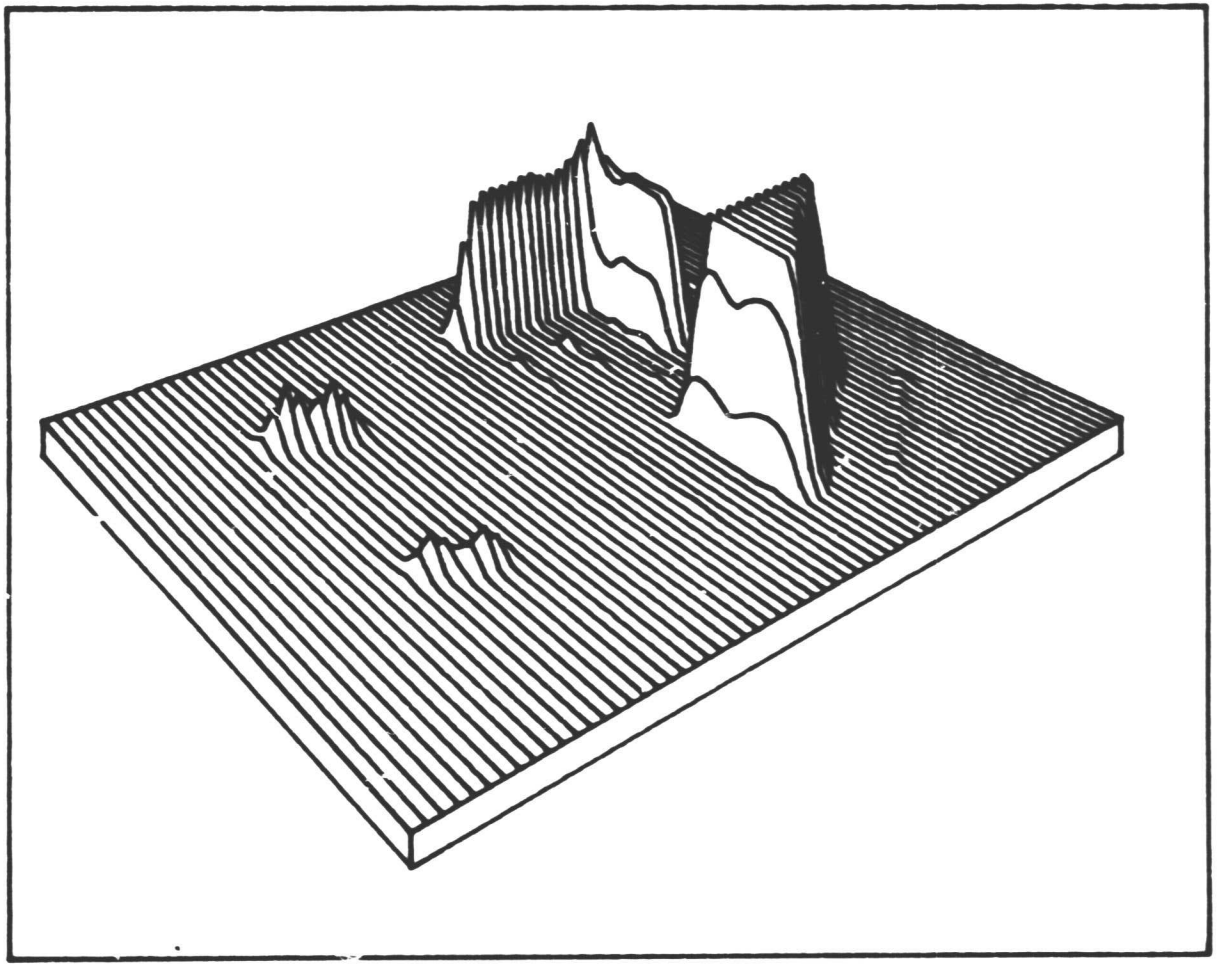
Run 2 : g(x,y)

Figure 4.15



Run 2 : log(1+g(x,y))

Figure 4.16



UNFOLDING ITERATION #10 - ALL CONSTRAINTS

Figure 4.17

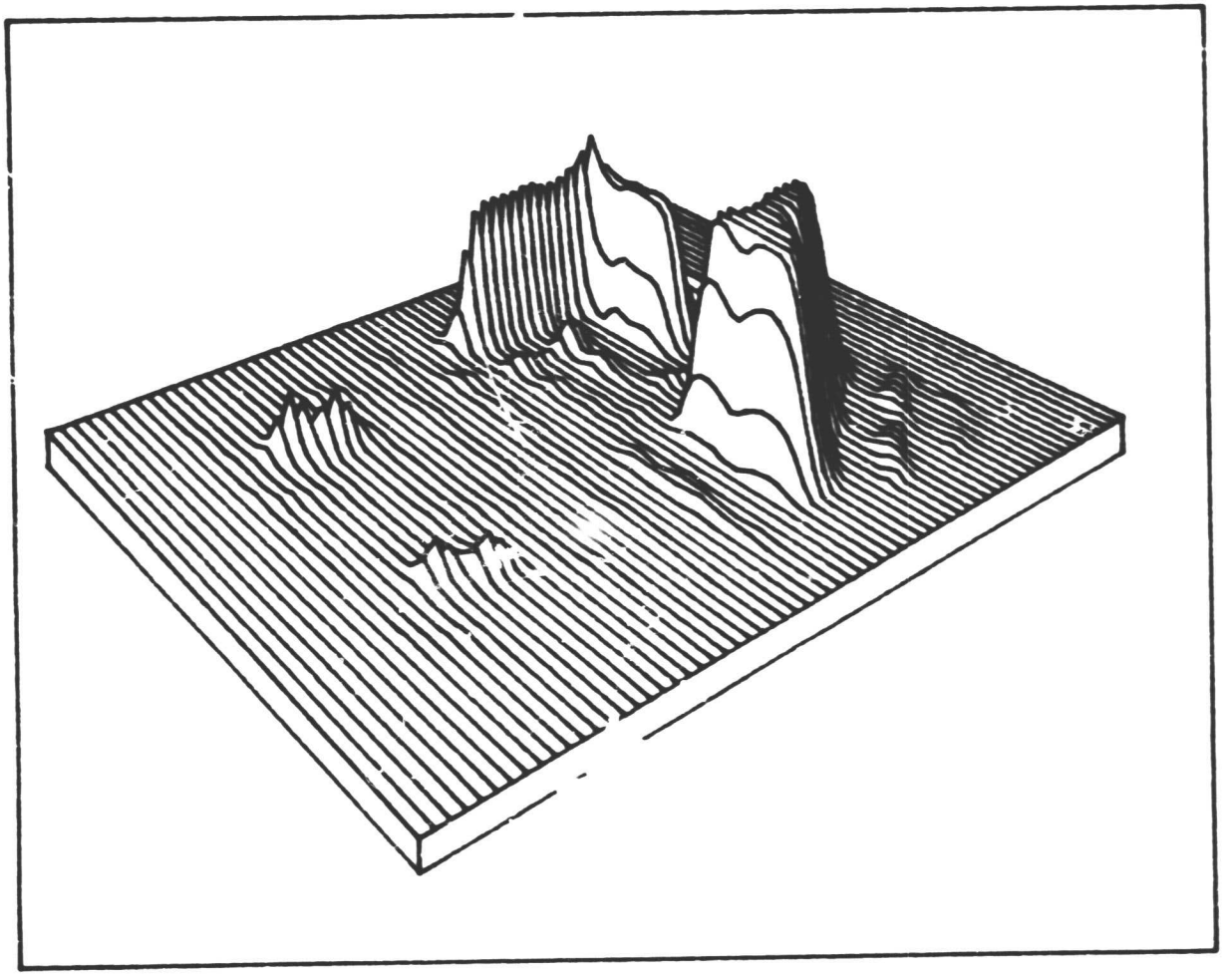
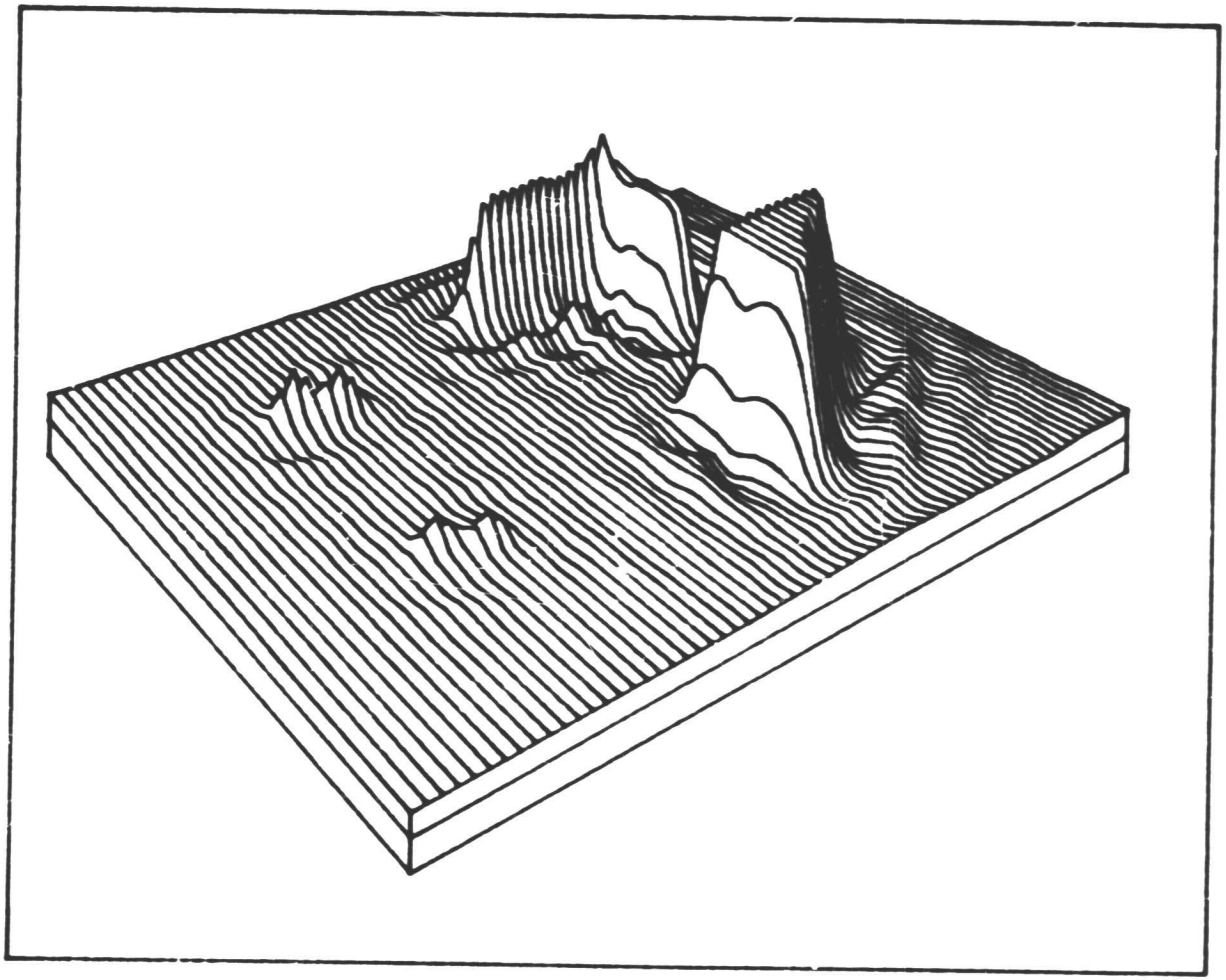
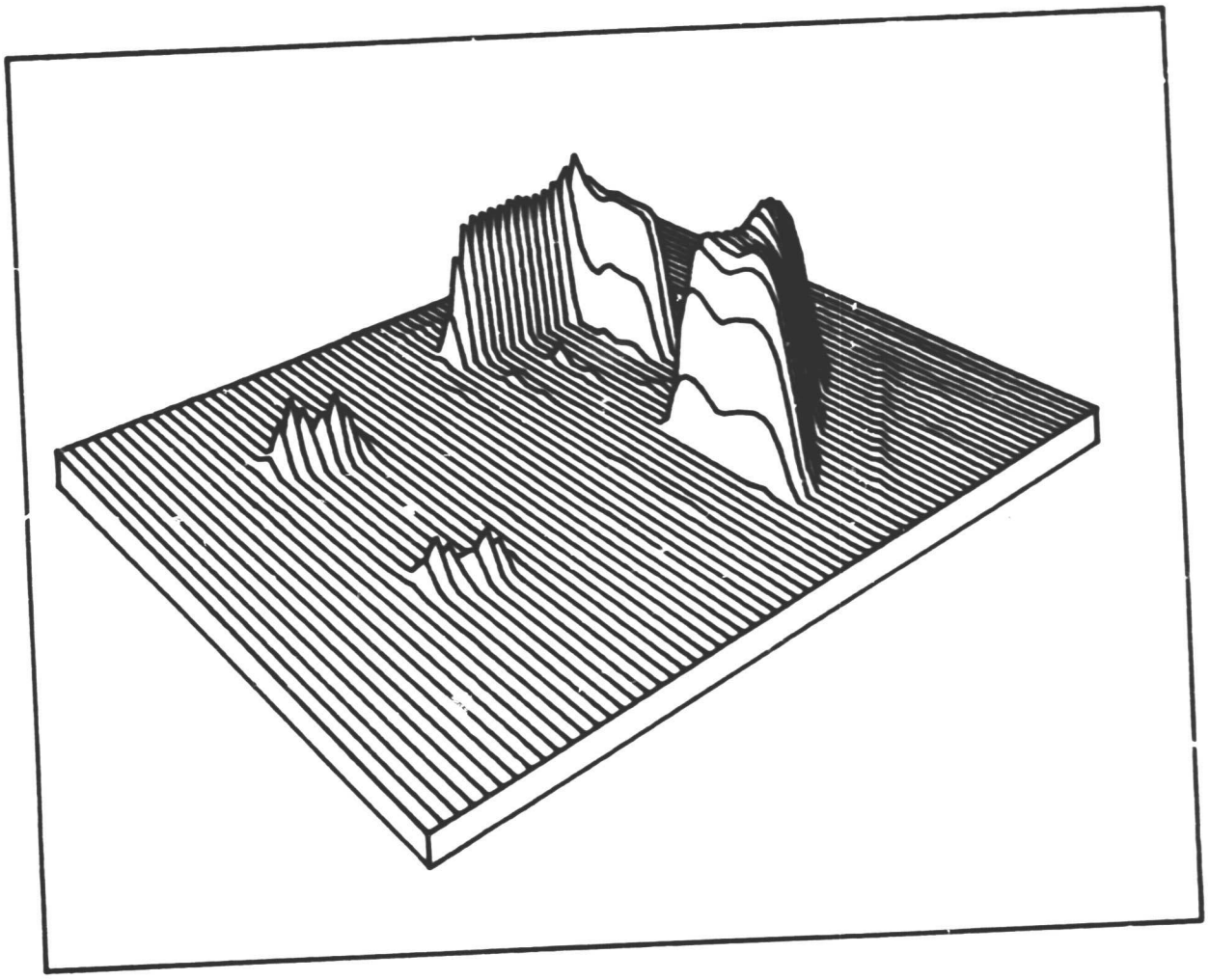


Figure 4.18



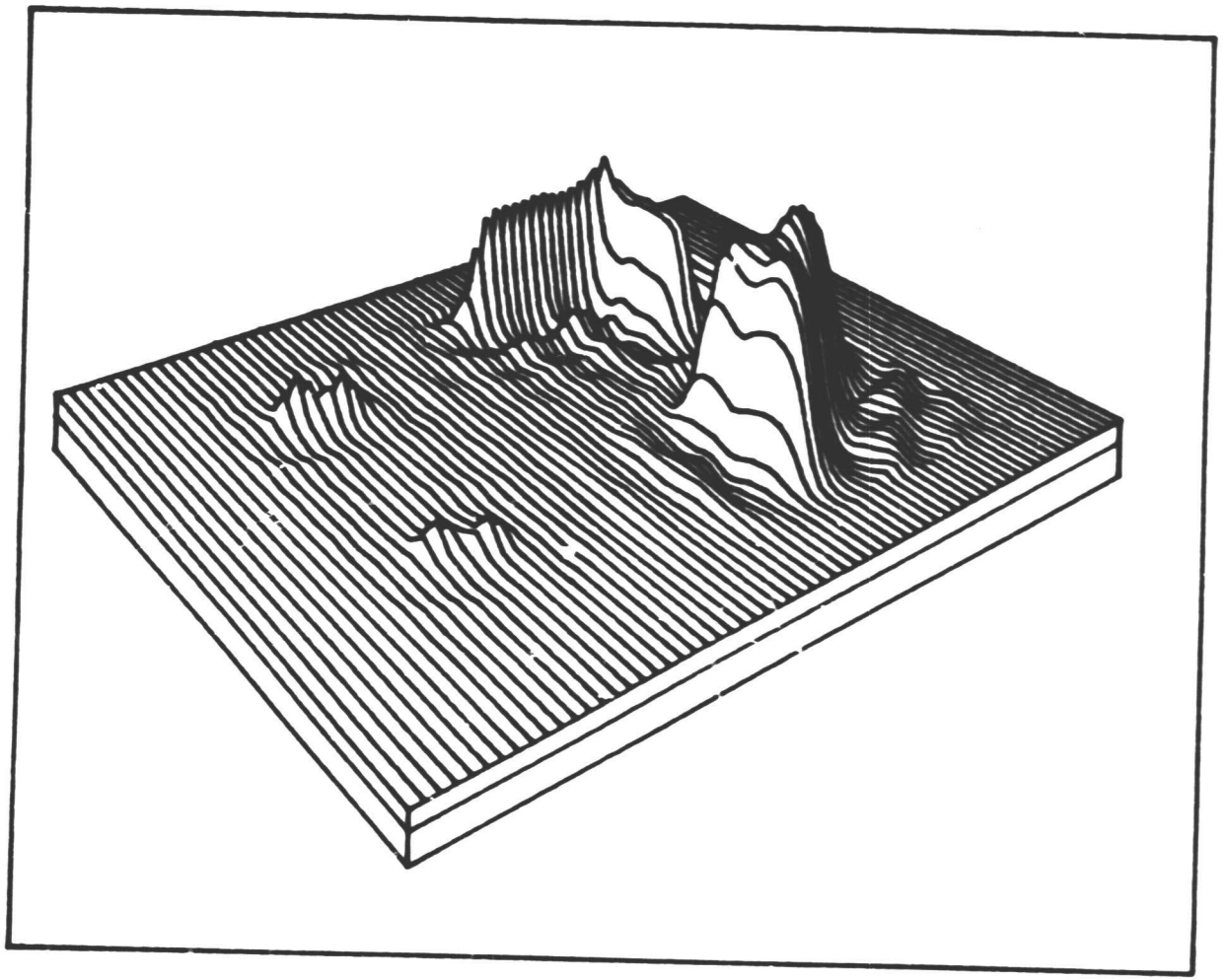
UNFOLDING ITERATION #10 - UPPER LIMIT CONSTRAINT

Figure 4.19



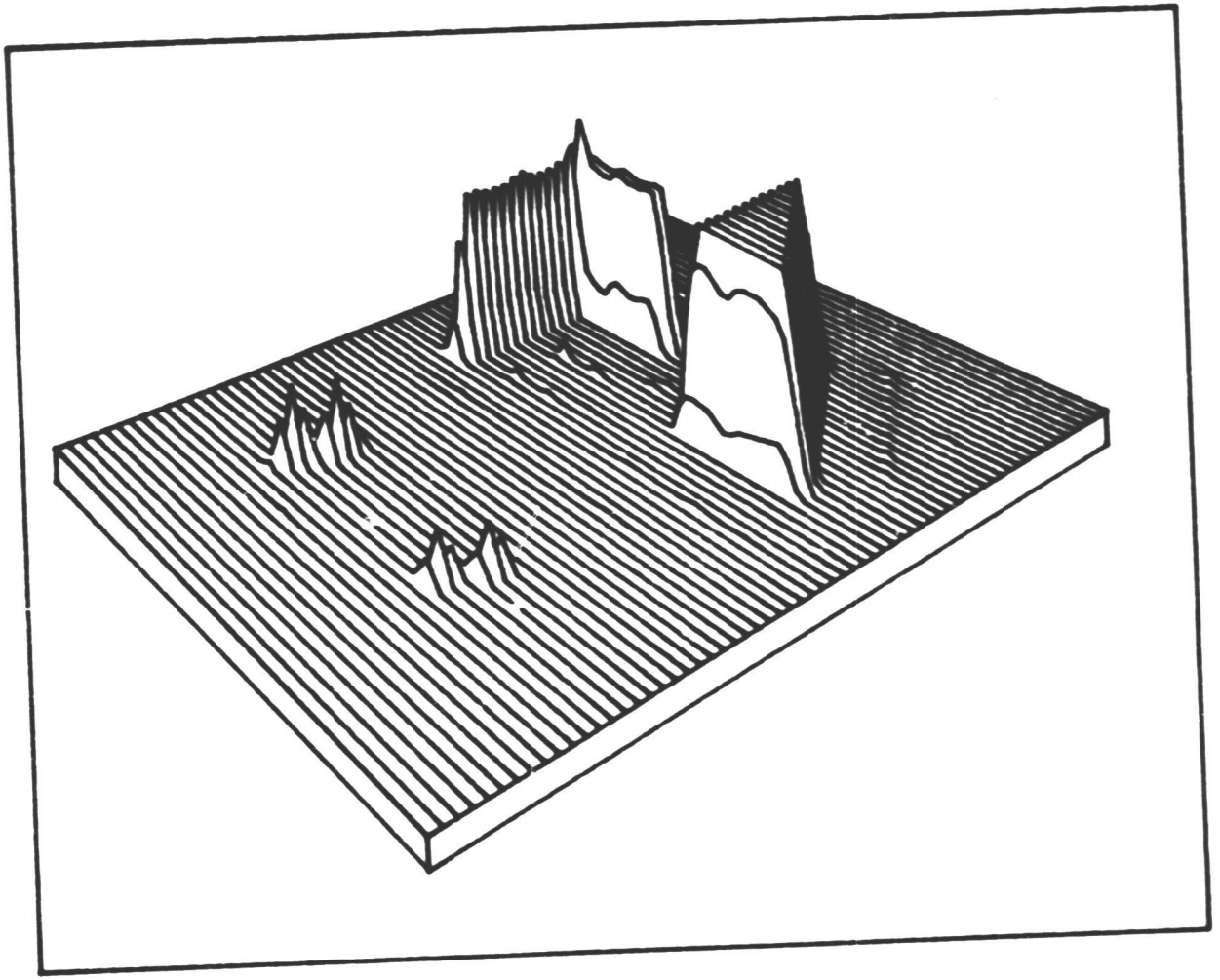
UNFOLDING ITERATION #10 - LOWER LIMIT CONSTRAINT

Figure 4.20



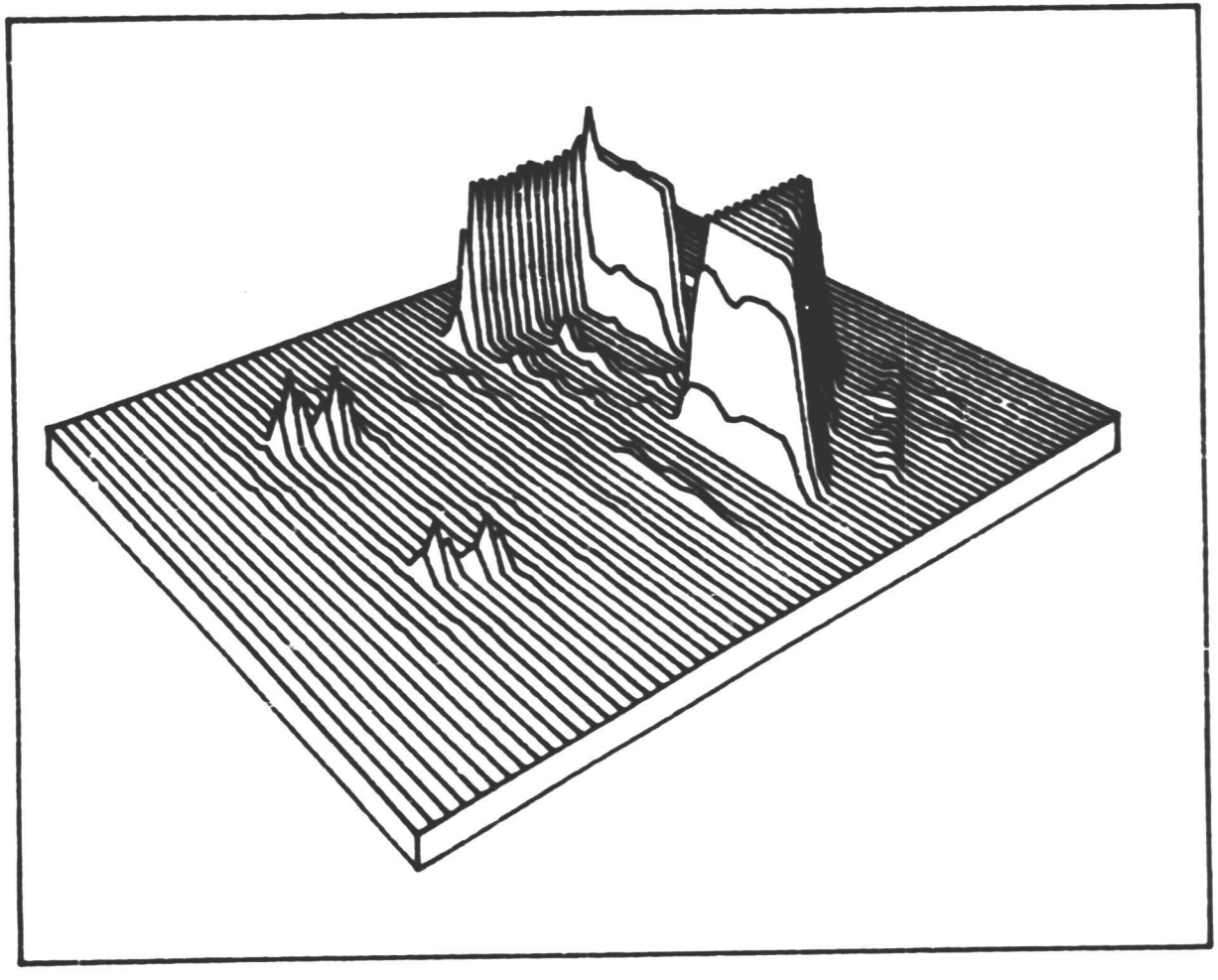
UNFOLDING ITERATION #10 - FINITE EXTENT CONSTRAINT

Figure 4.21



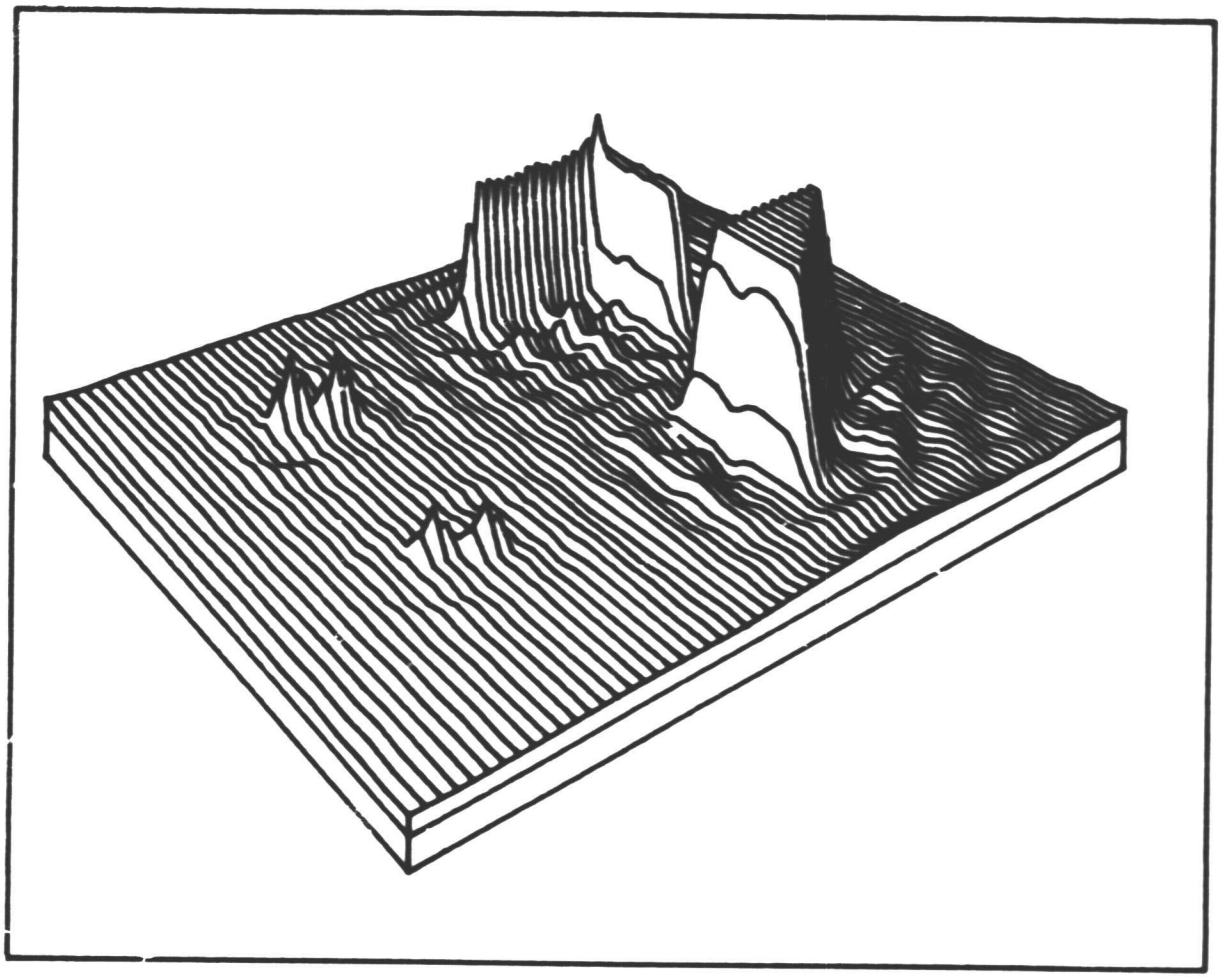
UNFOLDING ITERATION #20 - ALL CONSTRAINTS

Figure 4.22



UNFOLDING ITERATION 2X10 - ALL CONSTRAINTS

Figure 4.23



UNFOLDING ITERATION #20 - UPPER LIMIT CONSTRAINT

Figure 4.24

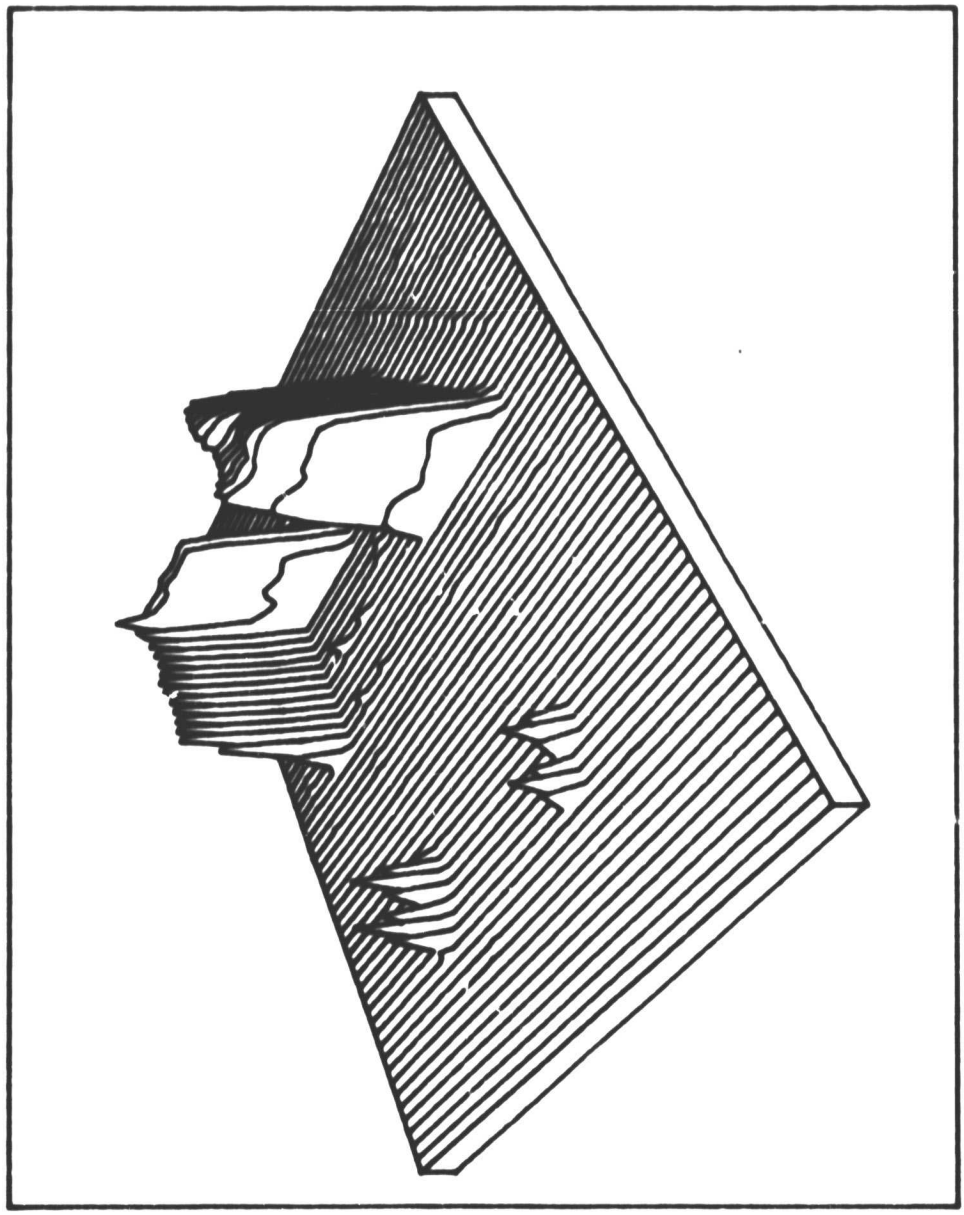


Figure 4.25

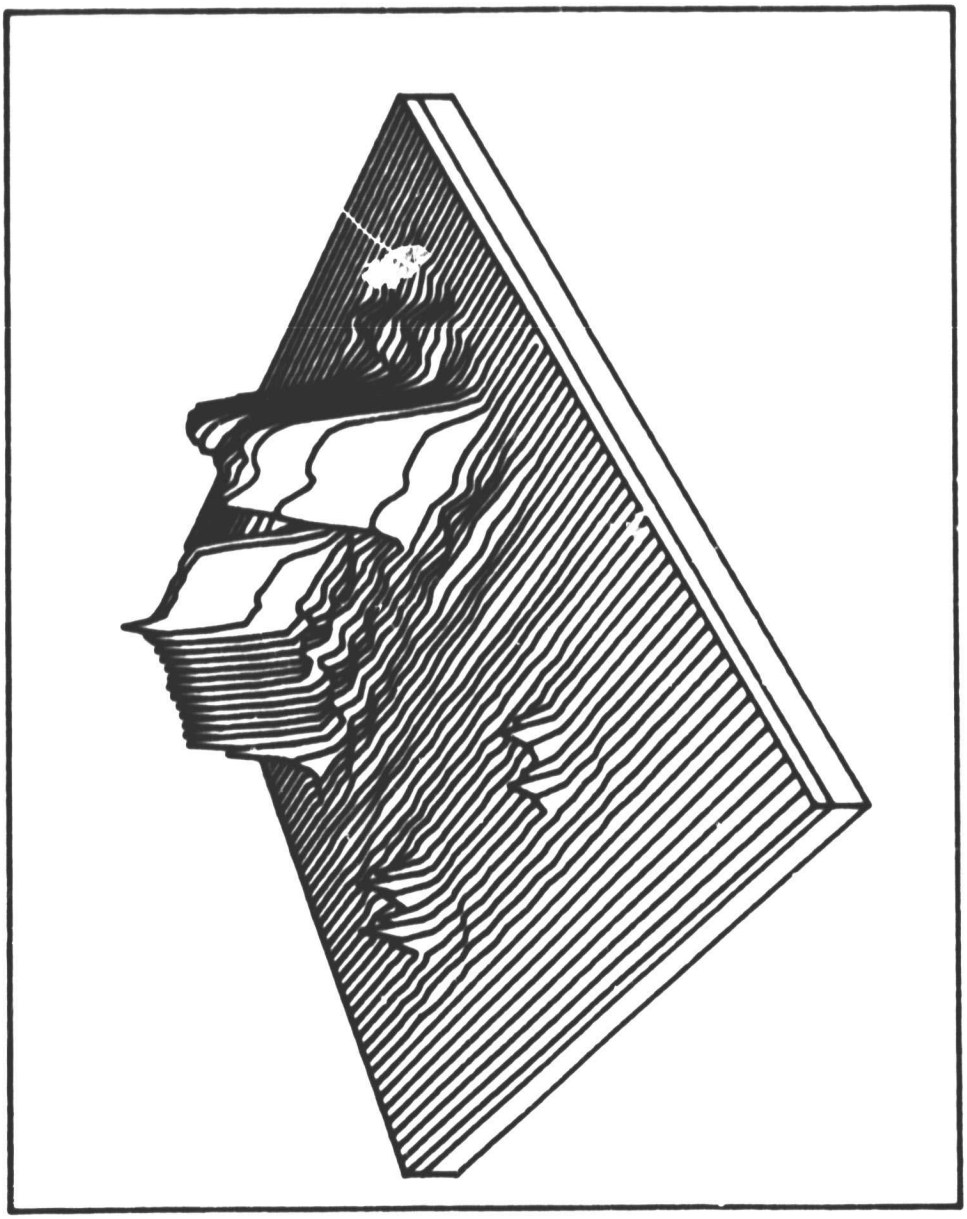
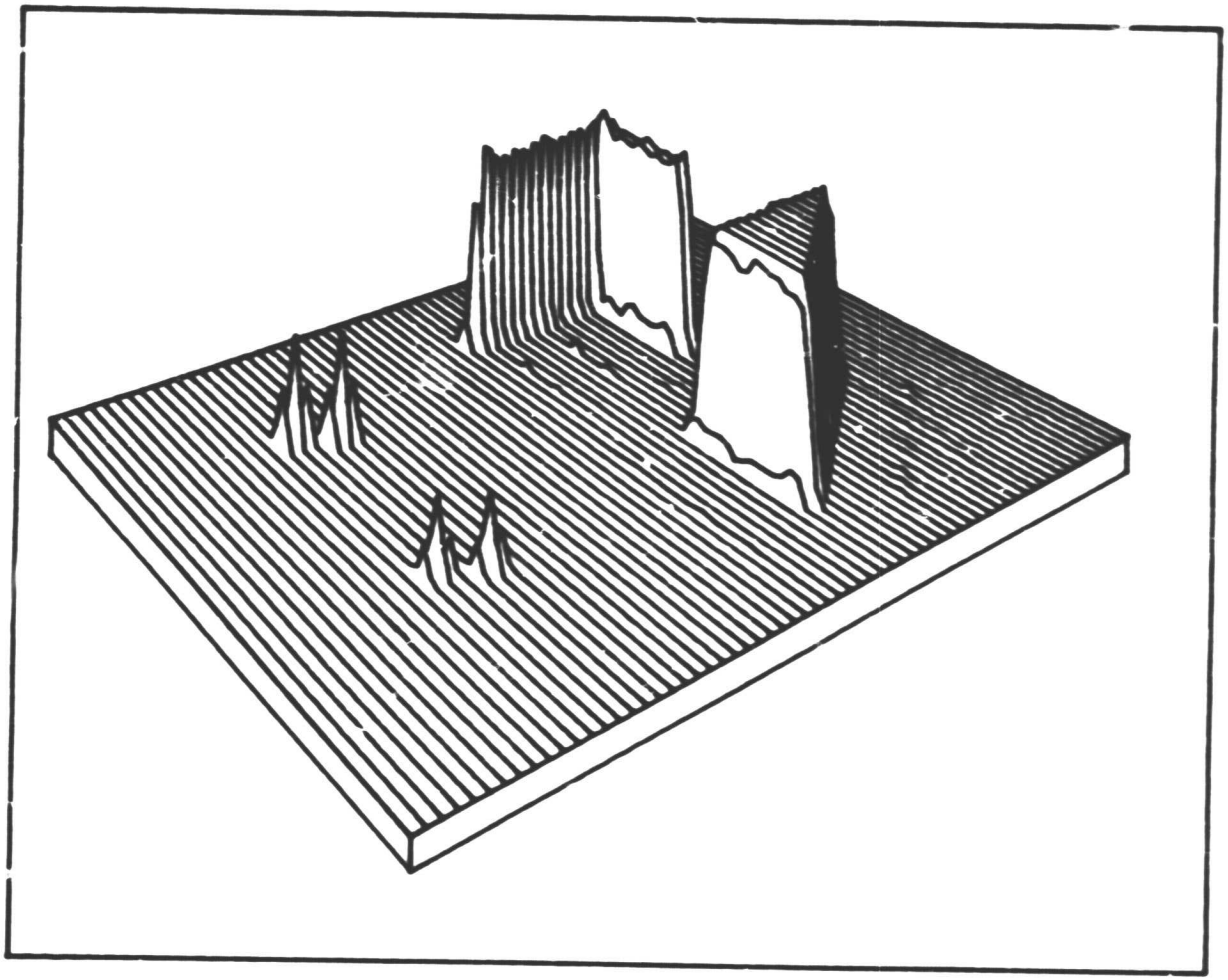


Figure 4.26



UNFOLDING ITERATION #80 - ALL CONSTRAINTS

Figure 4.27

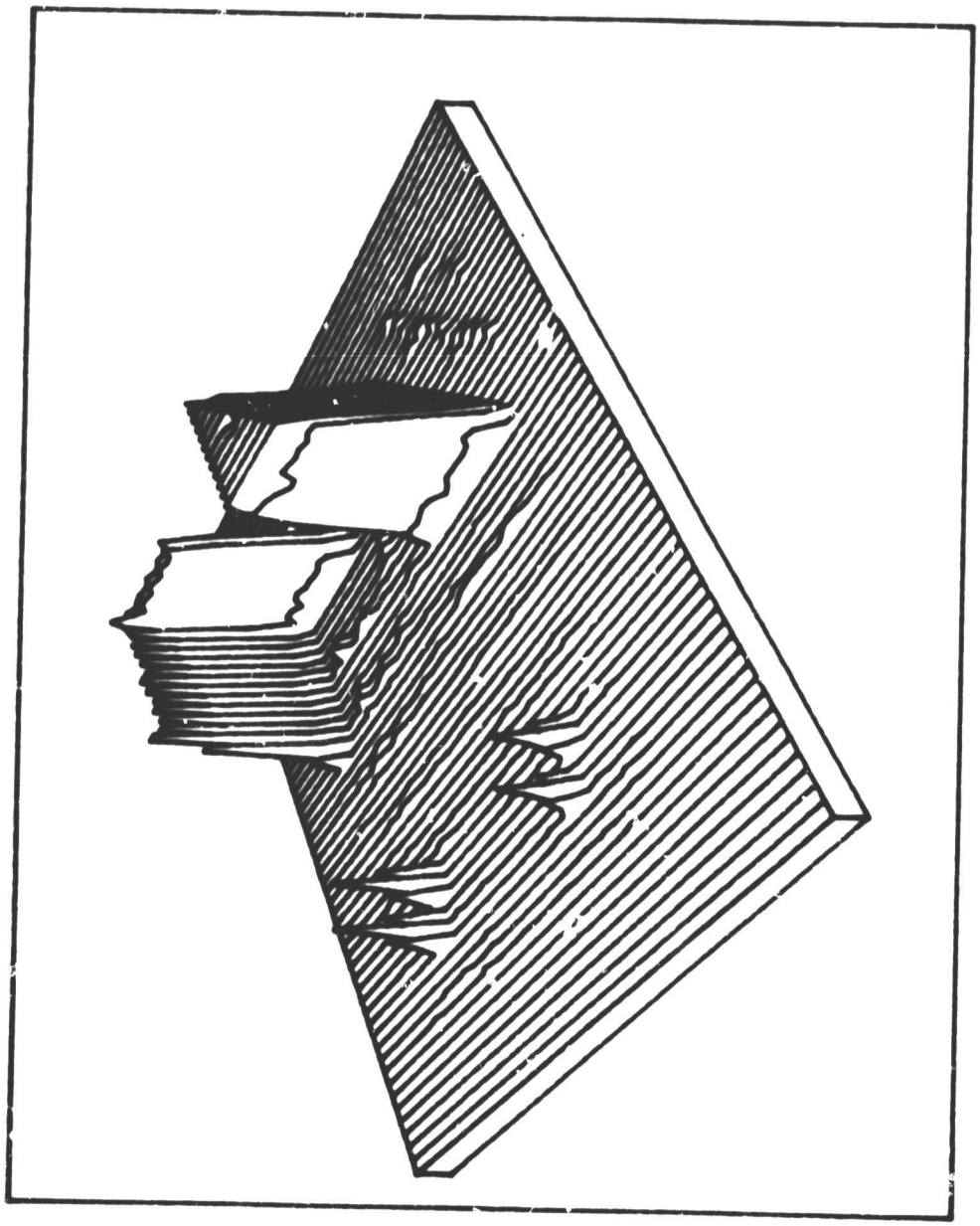
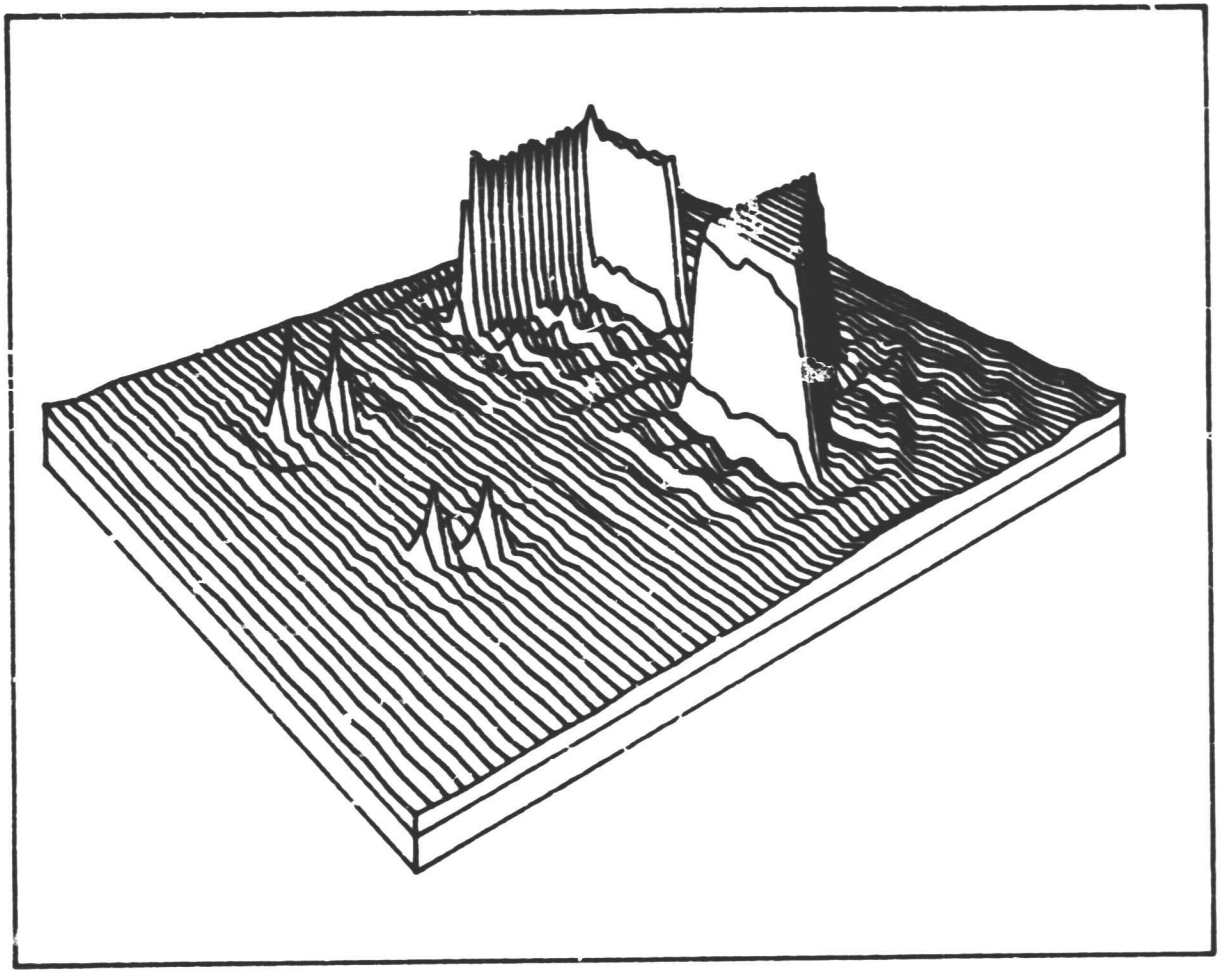
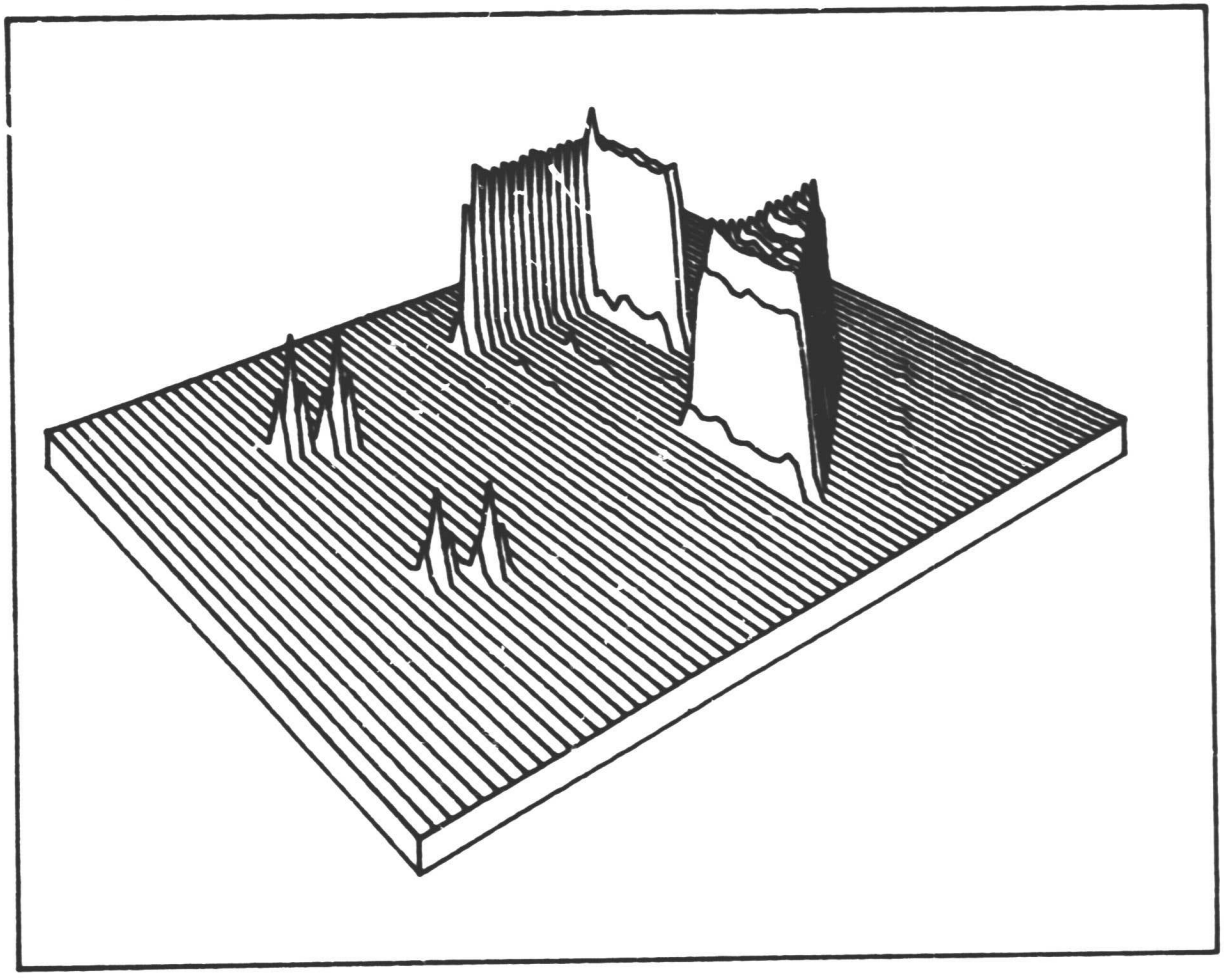


Figure 4.28



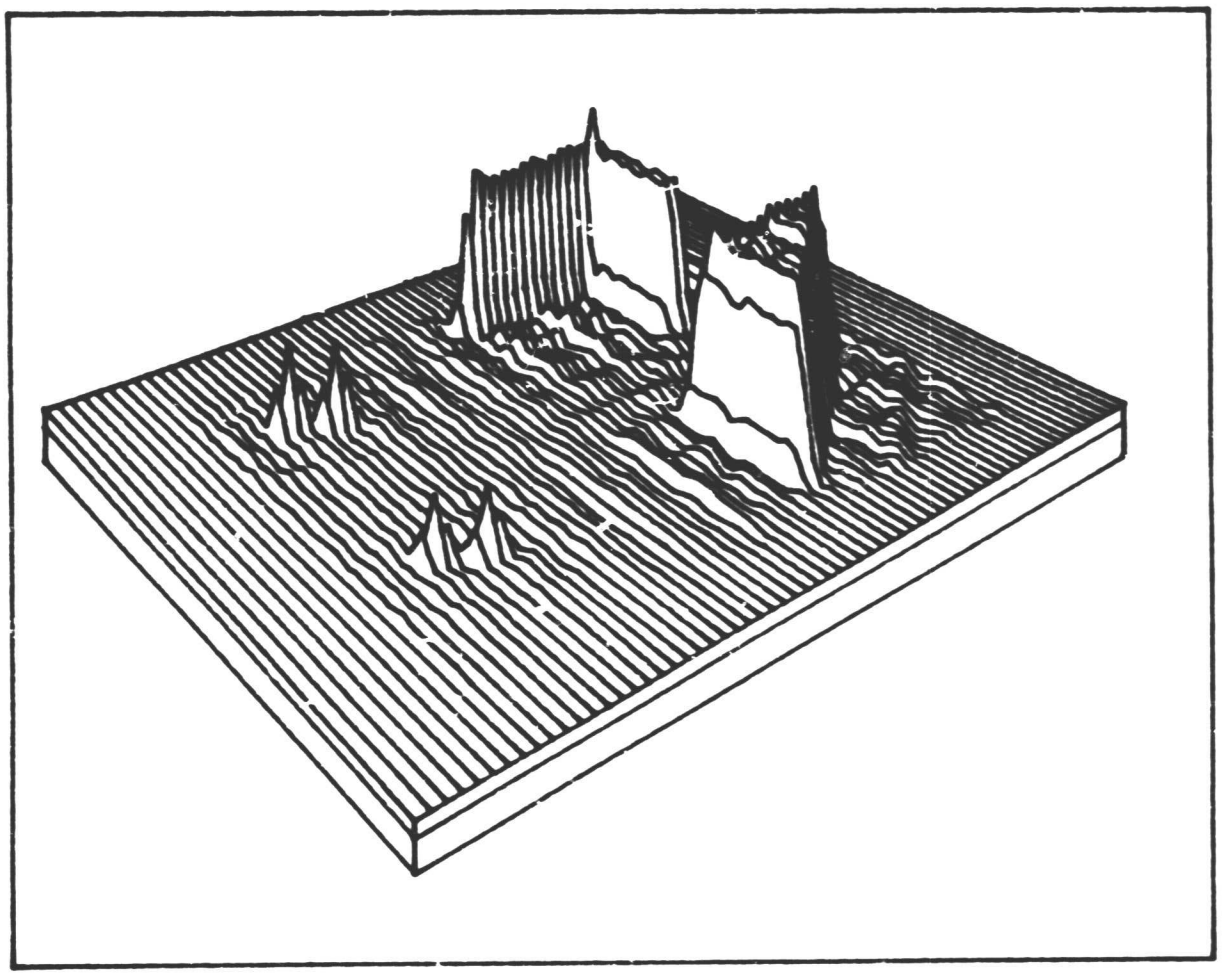
UNFOLDING ITERATION #80 - UPPER LIMIT CONSTRAINT

Figure 4.29



UNFOLDING ITERATION #50 - LOWER LIMIT CONSTRAINT

Figure 4.30



UNFOLDING ITERATION #80 - FINITE EXTENT CONSTRAINT

Figure 4.31

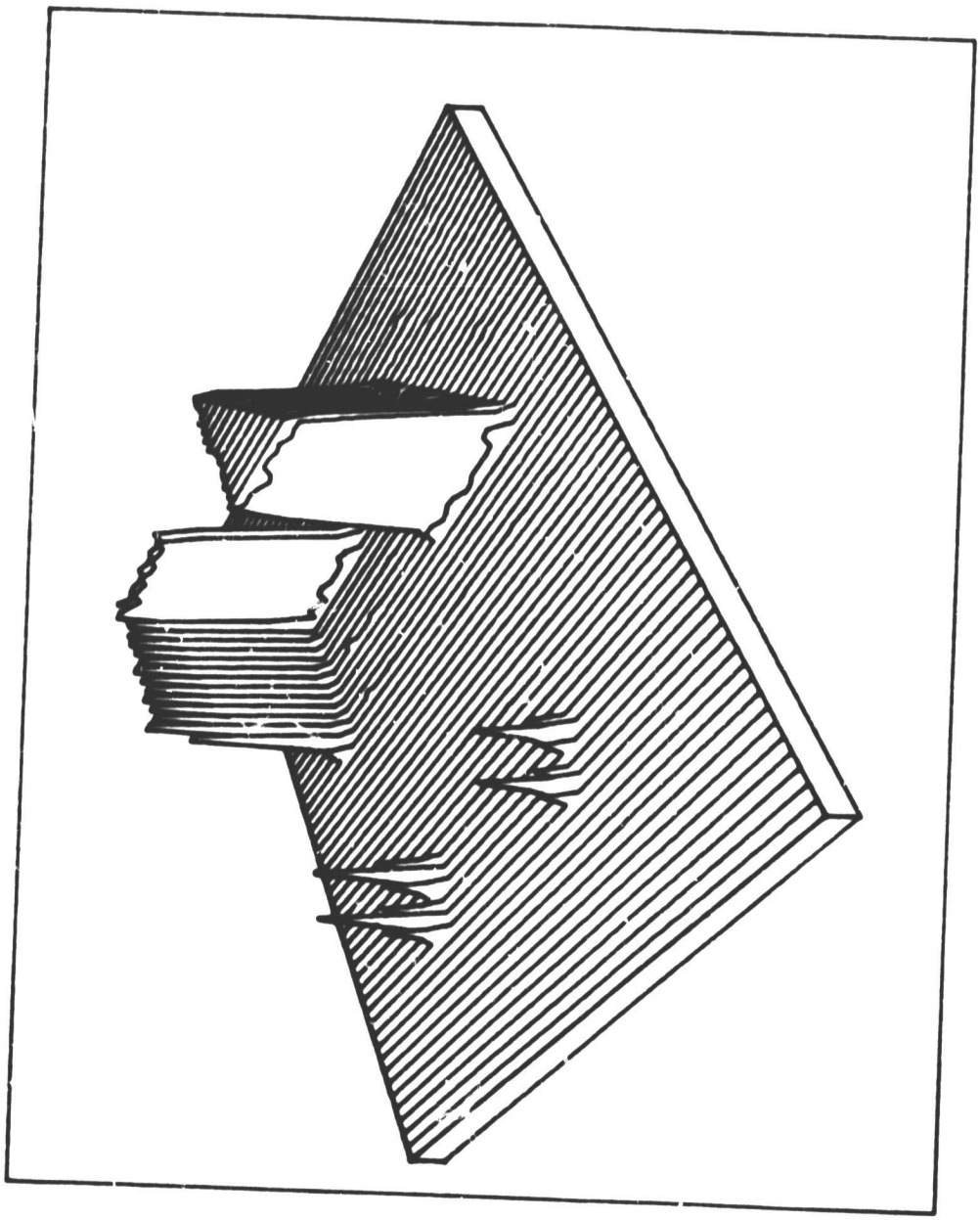
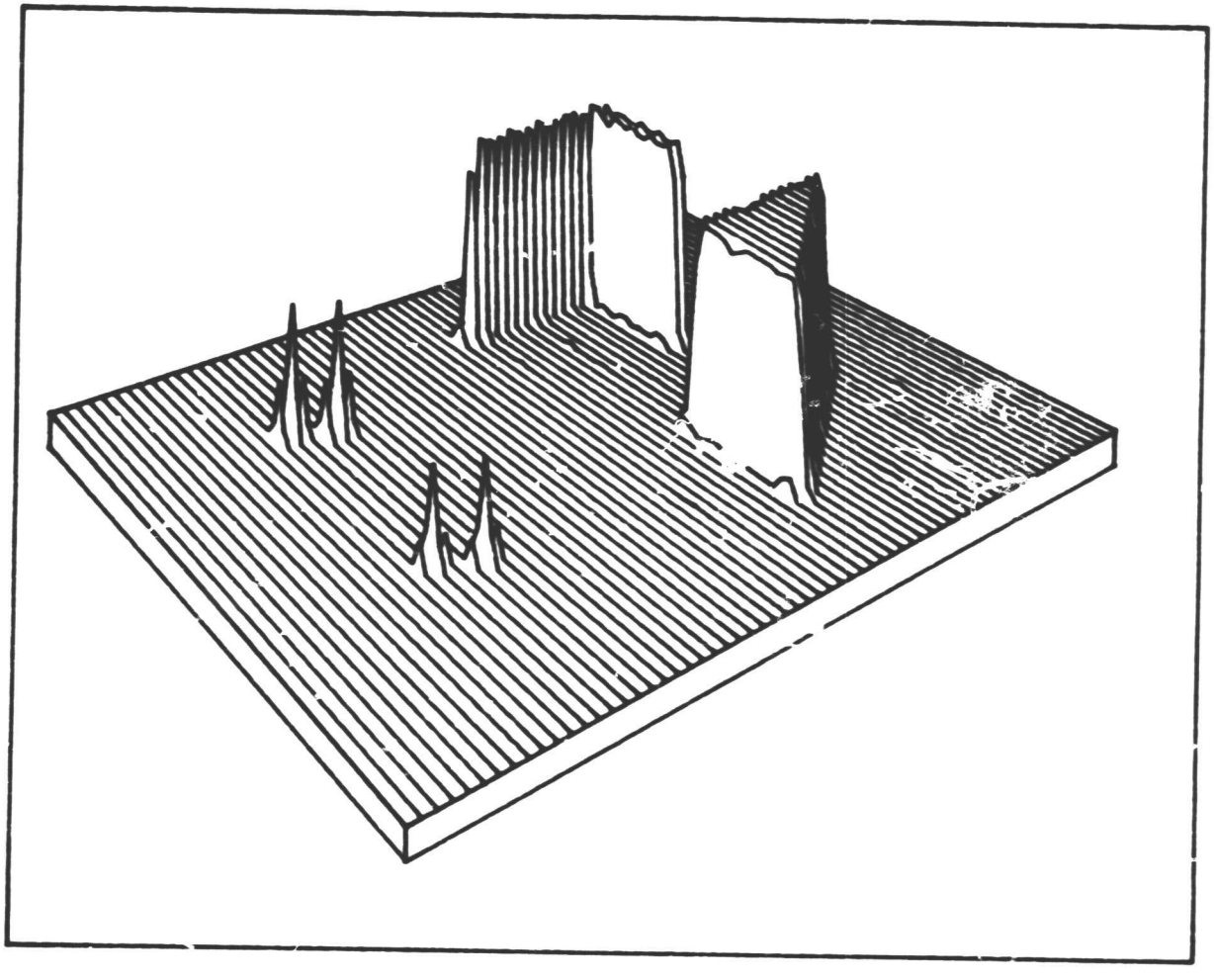
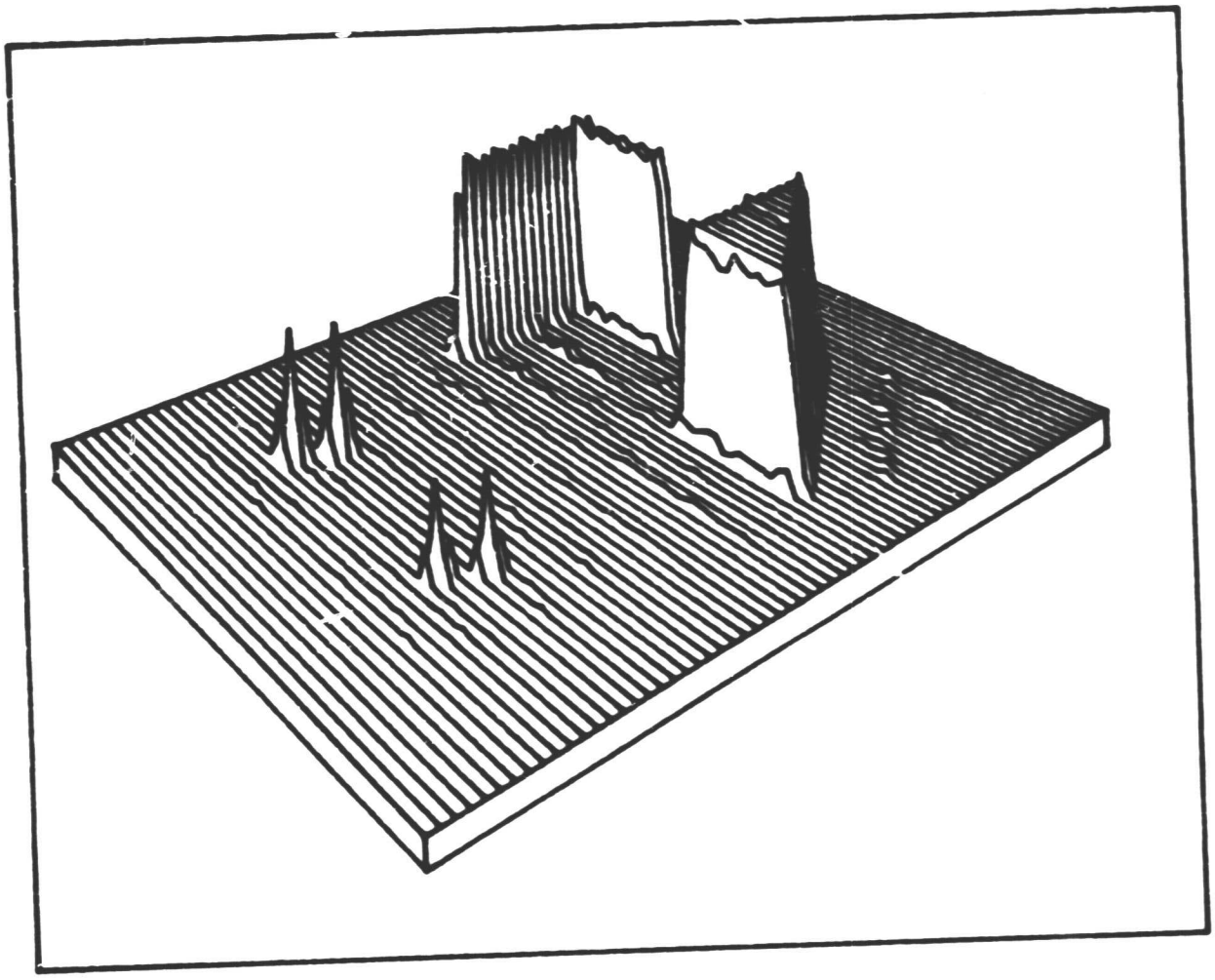


Figure 4.32



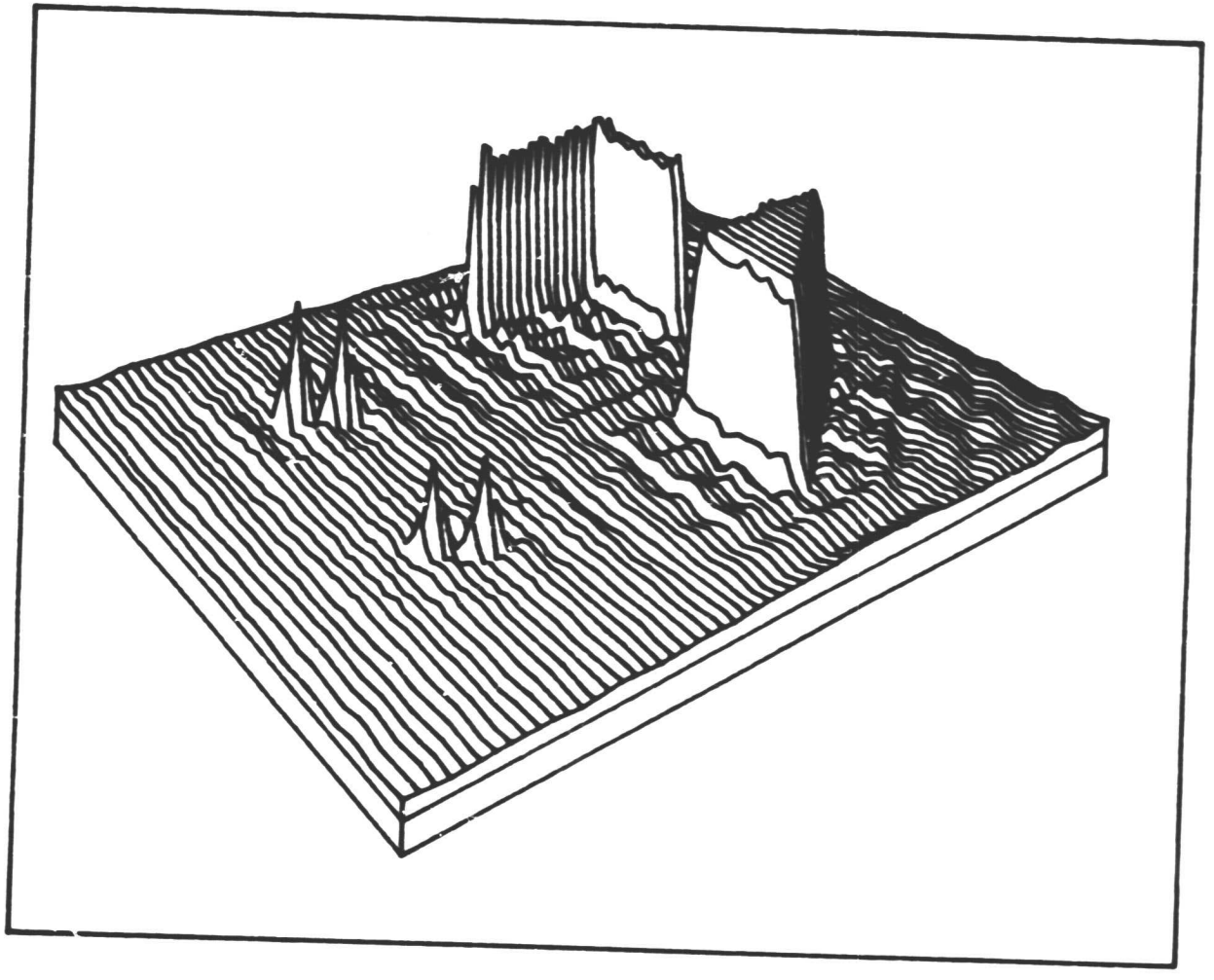
UNFOLDING ITERATION #100 - ALL CONSTRAINTS

Figure 4.33



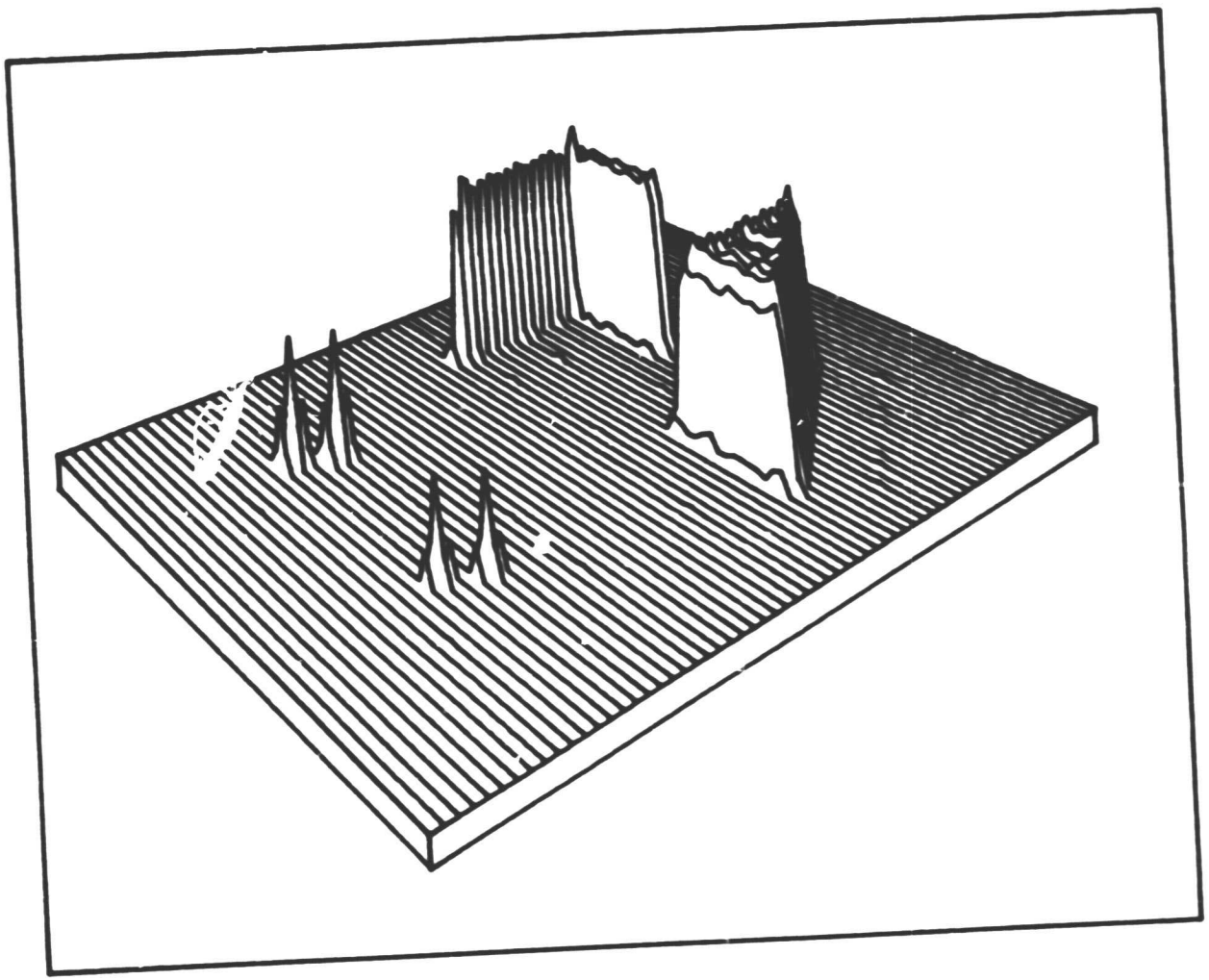
UNFOLDING ITERATION 10X10 - ALL CONSTRAINTS

Figure 4.34



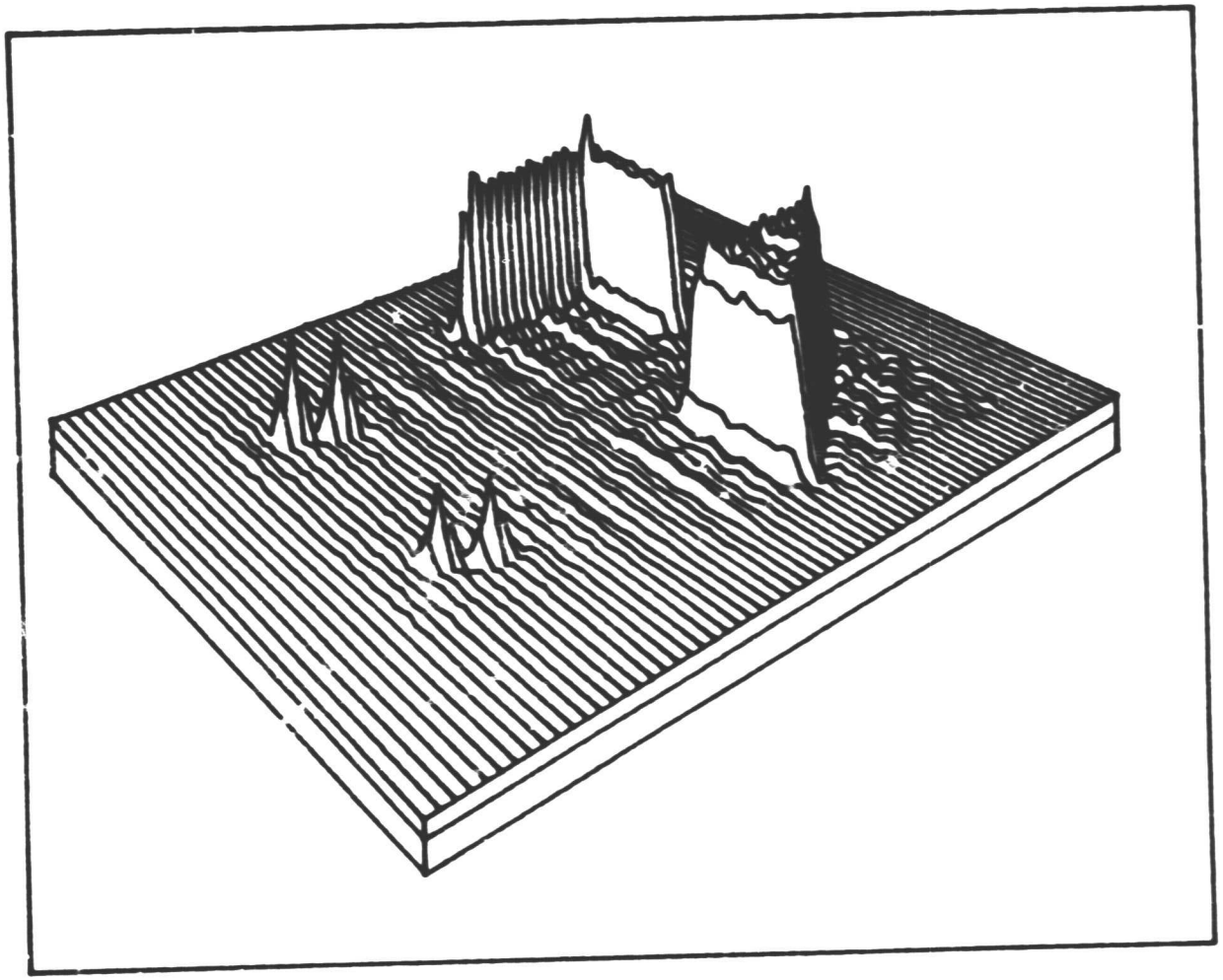
UNFOLDING ITERATION #100 - UPPER LIMIT CONSTRAINT

Figure 4.35



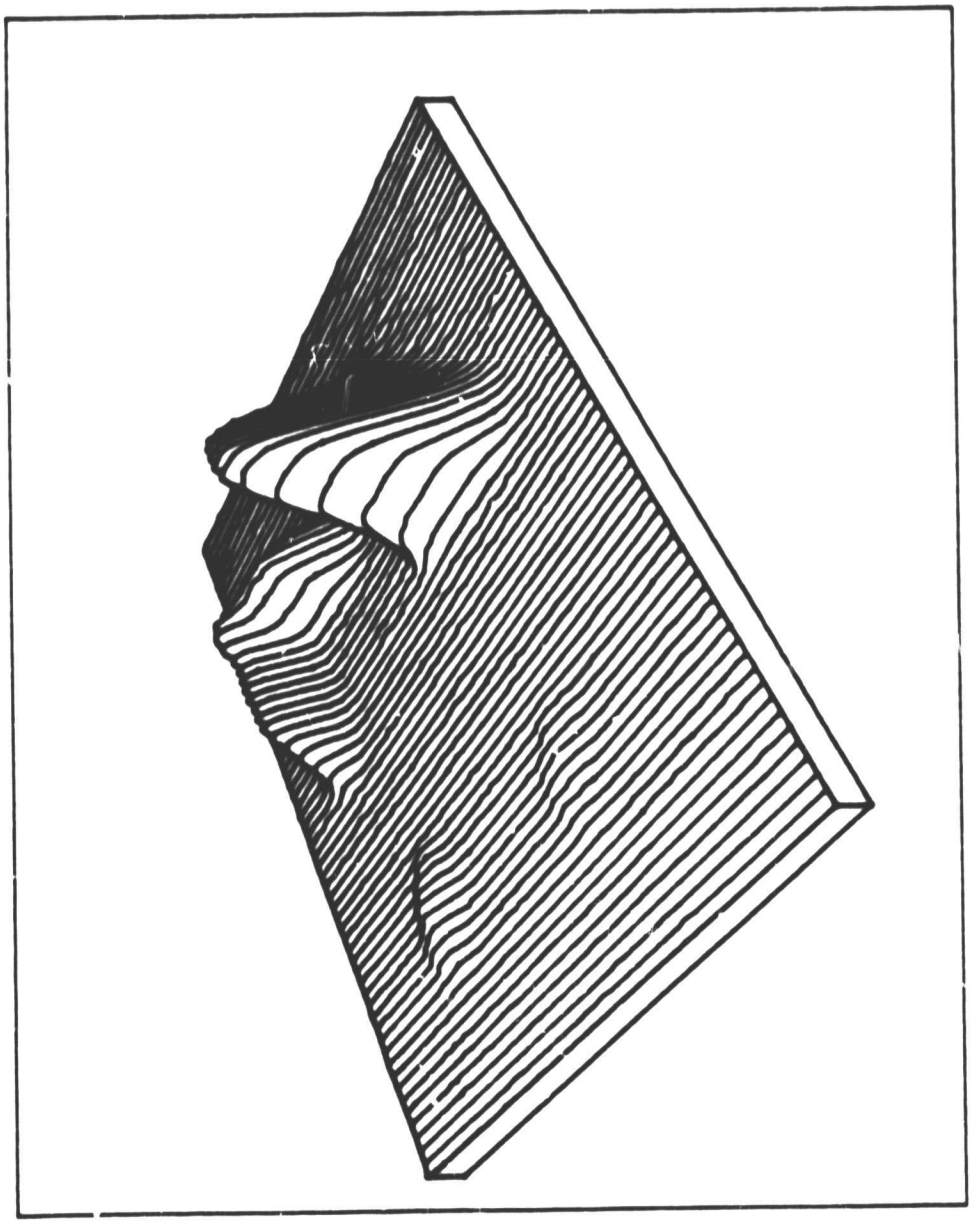
UNFOLDING ITERATION #100 - LOWER LIMIT CONSTRAINT

Figure 4.36



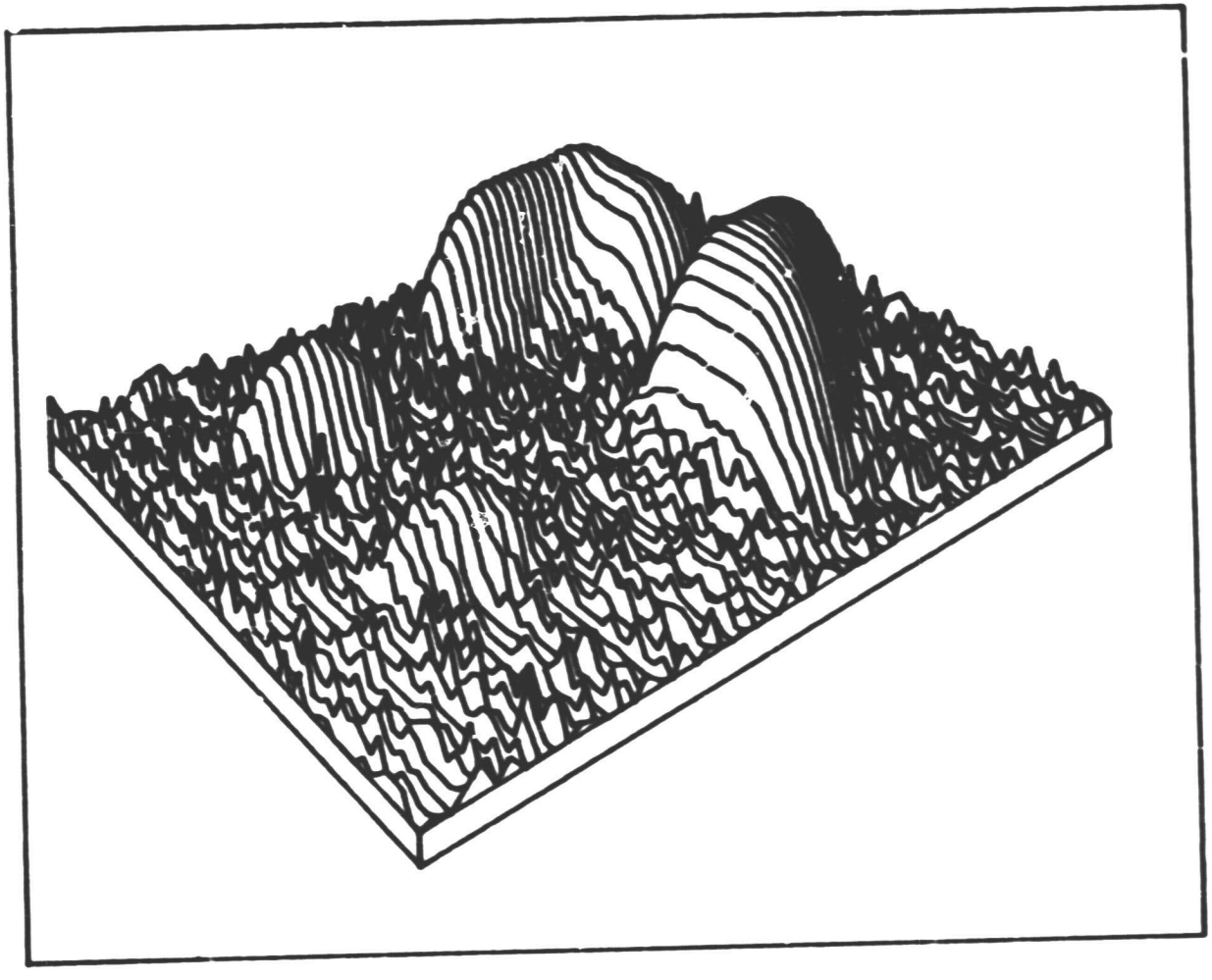
UNFOLDING ITERATION #100 - FINITE EXTENT CONSTRAINT

Figure 4.37



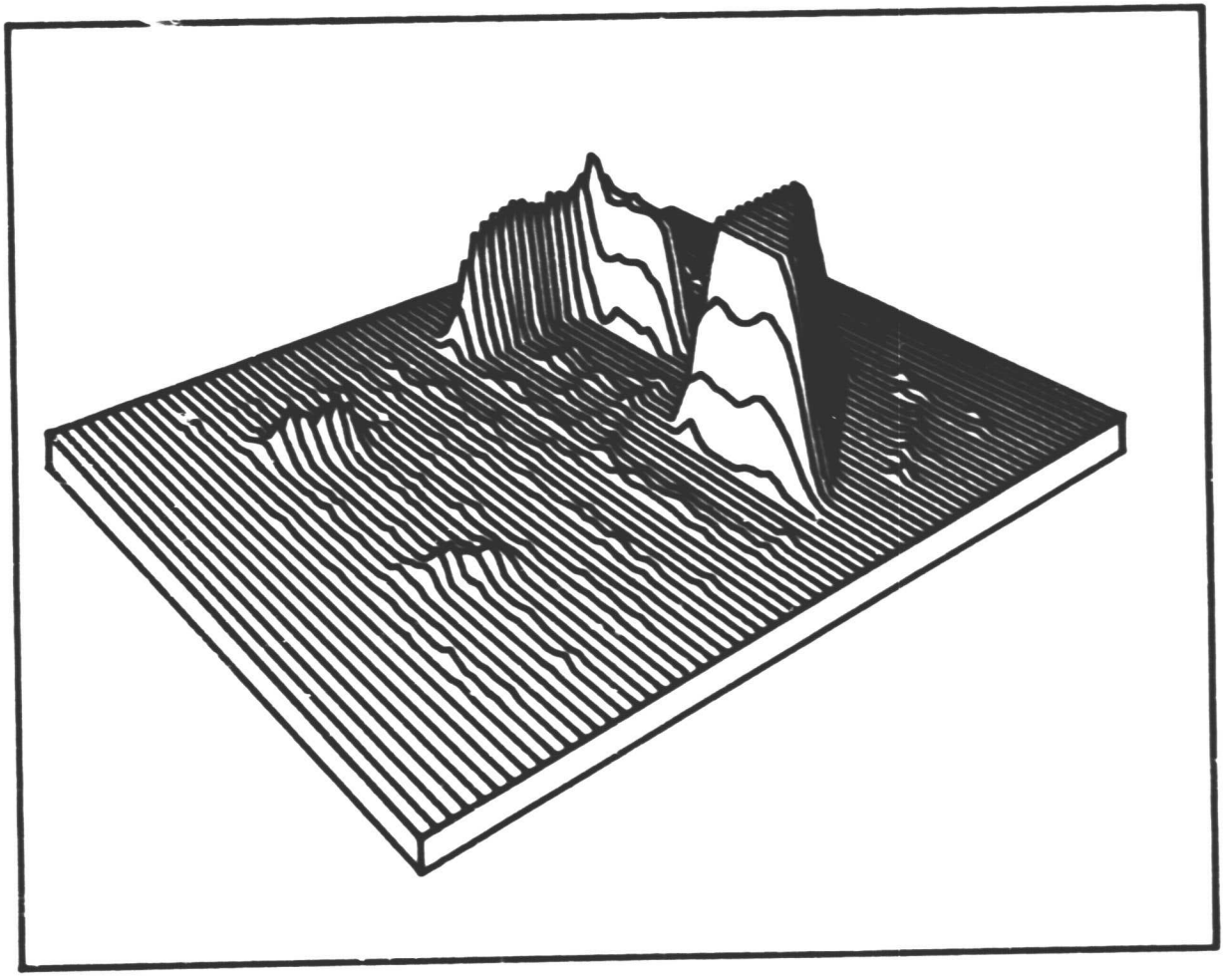
g(x,y) + Gaussian noise ; S/N=200

Figure 4.38



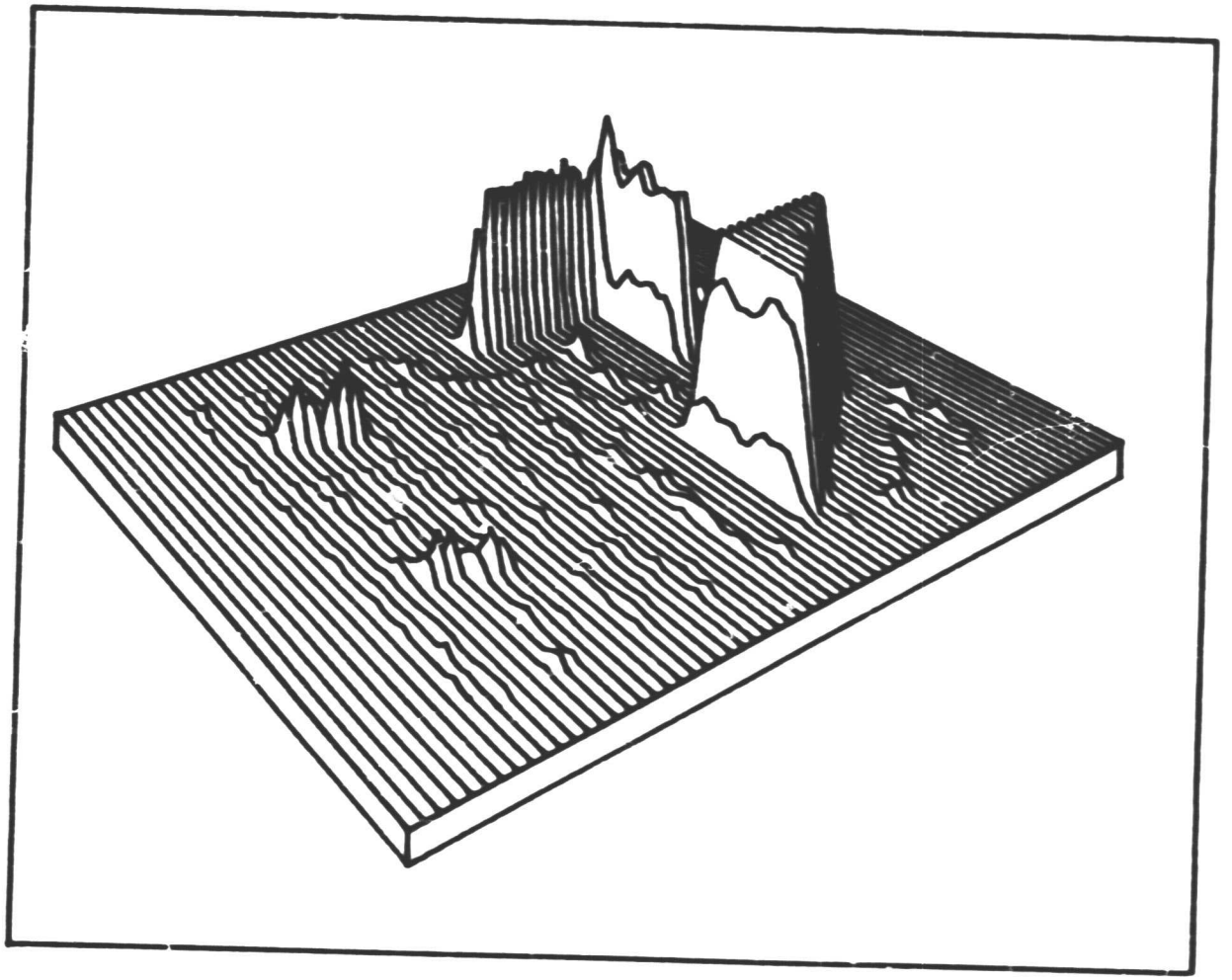
log(1+g(x,y)) ; S/N=200

Figure 4.39



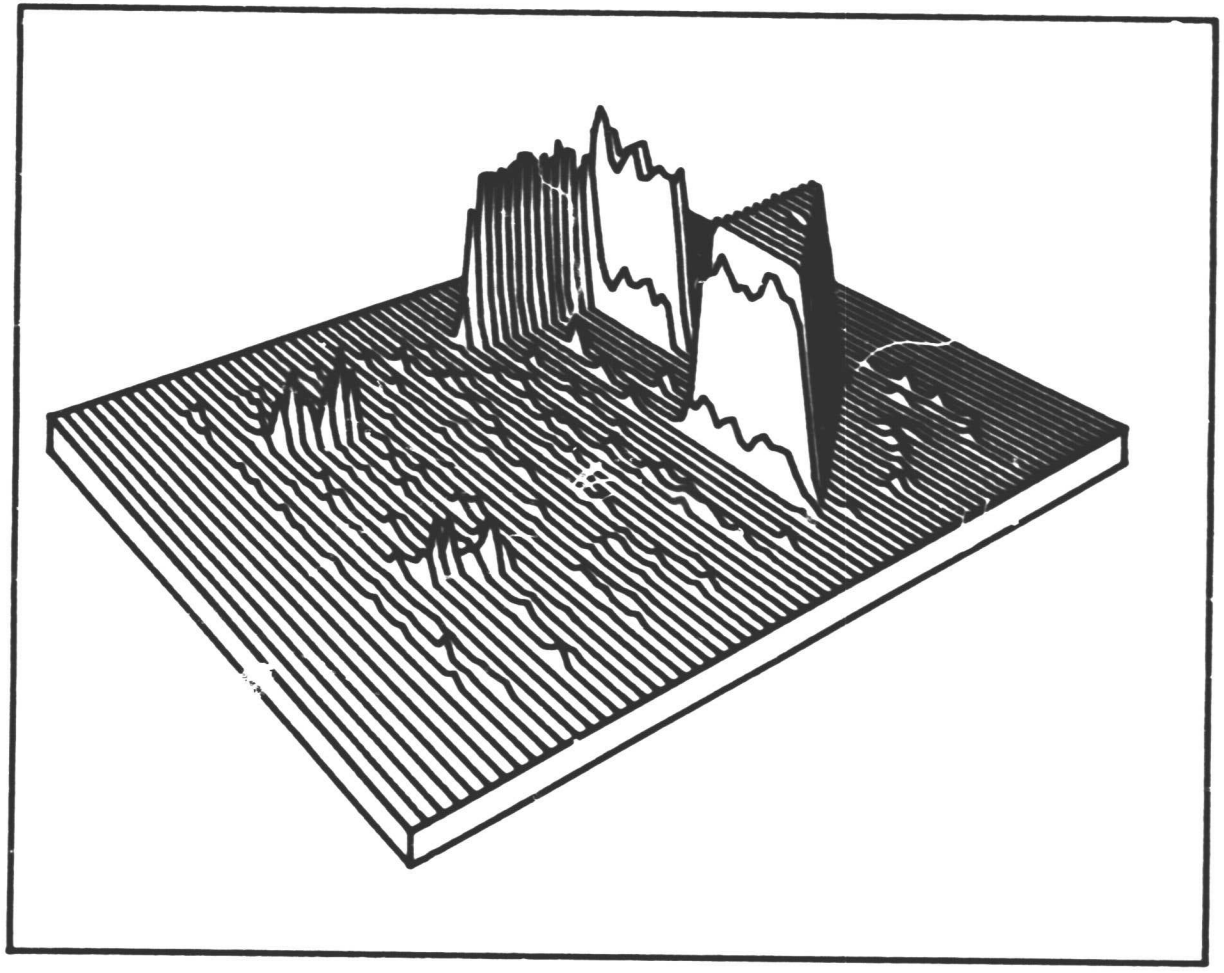
UNFOLDING ITERATION 1002+1005+1010 - 5 SMOOTHINGS - SMP=200

Figure 4.40



UNFOLDING ITERATION 10X.5 + 10X4 - 20 SMOOTHINGS - SNR=200

Figure 4.41



UNFOLDING ITERATION 103.5 + 25x4 - 20 SMOOTHINGS - SNR=200

Figure 4.42

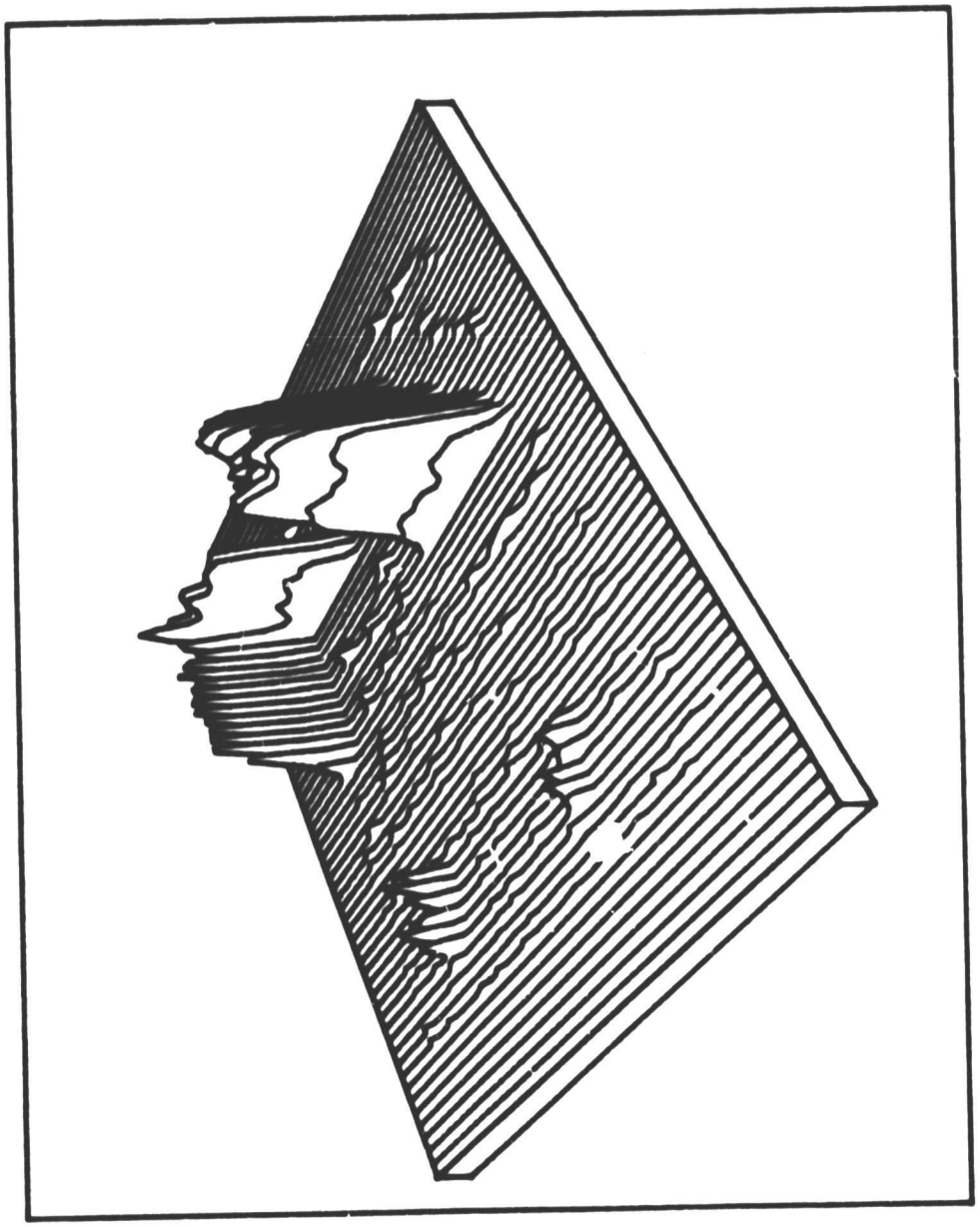


Figure 4.43

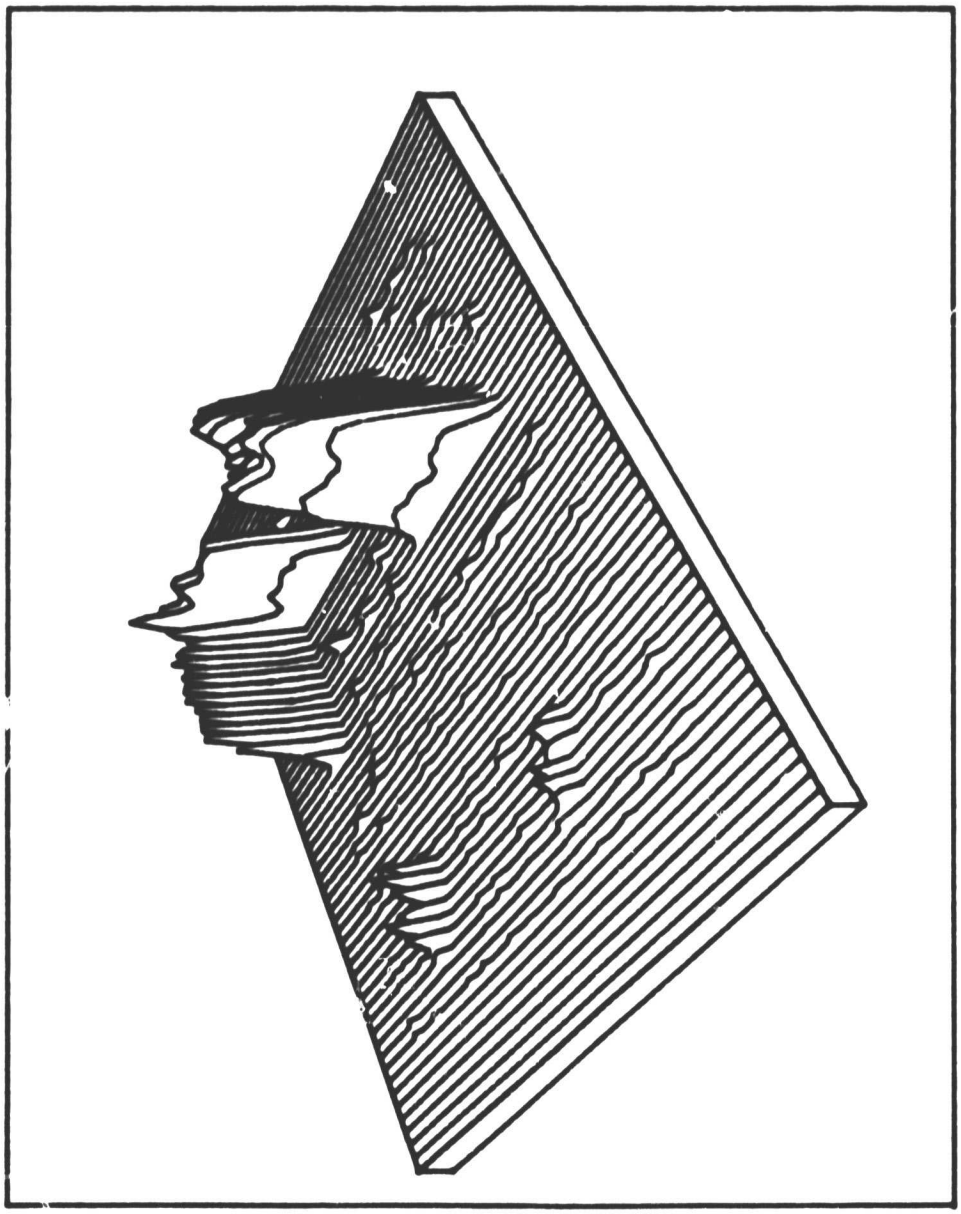
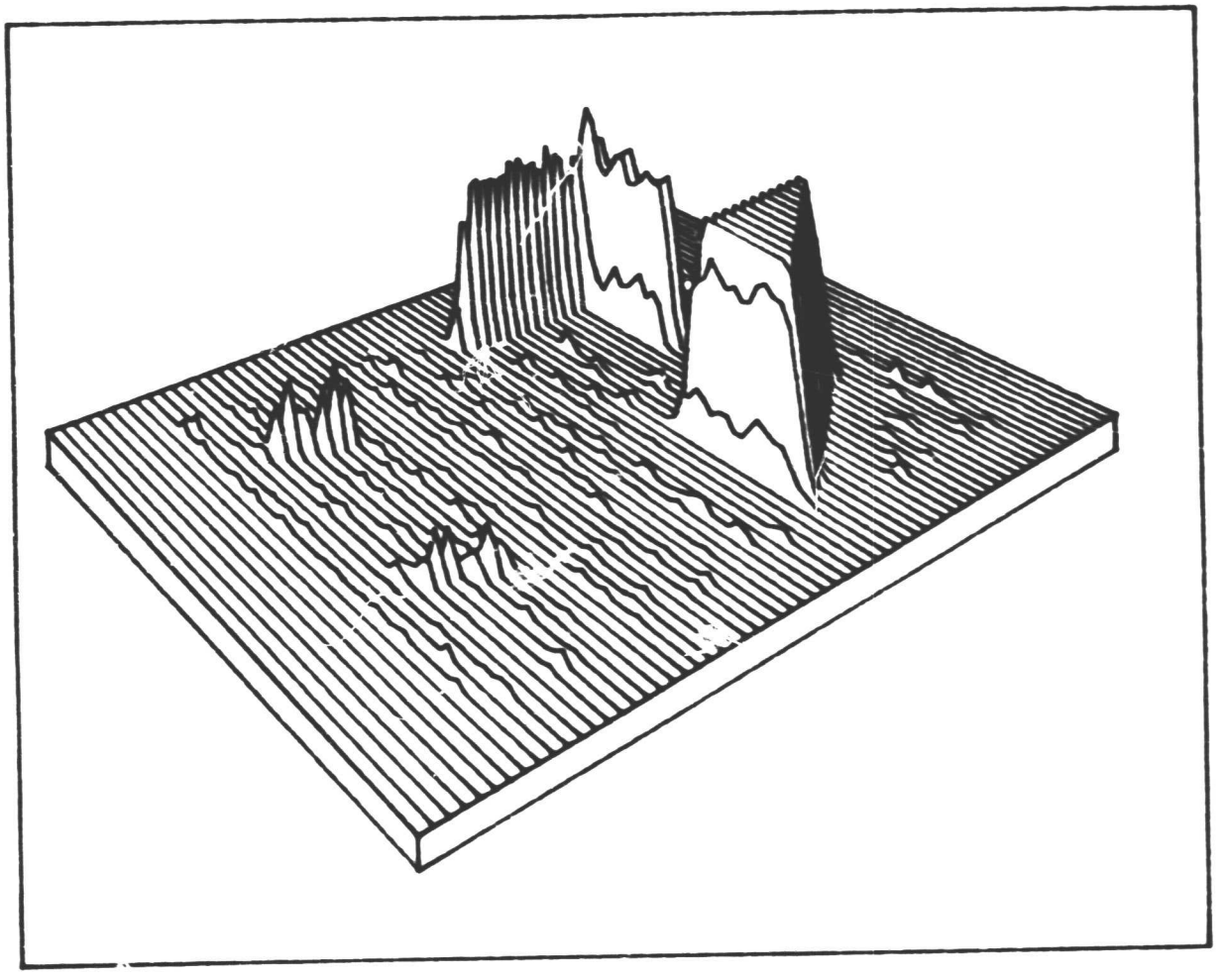
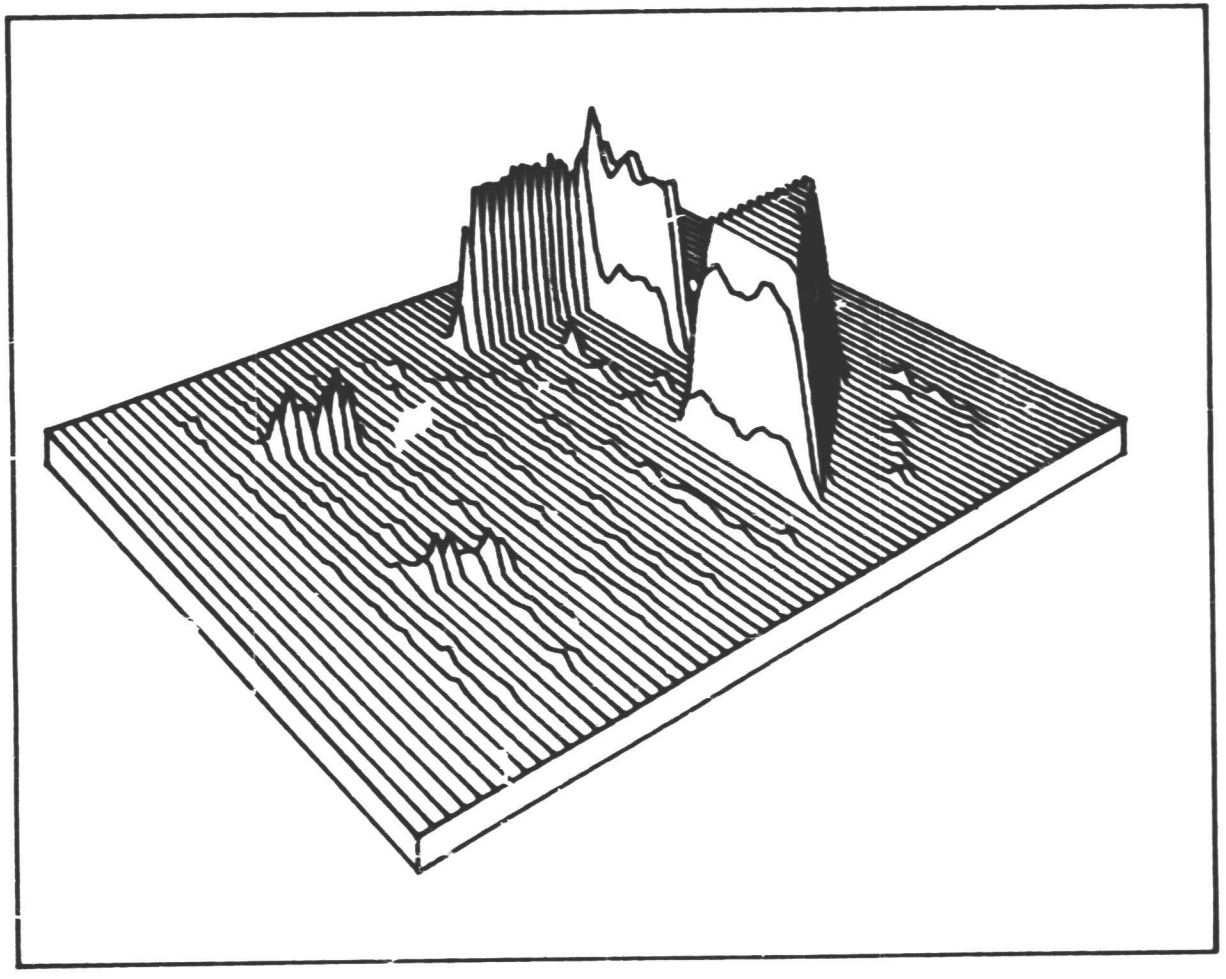

UNFOLDING ITERATION 2002 - 20 SMOOTHINGS - NO PENK CONSTRAINT - SNR=200

Figure 4.44



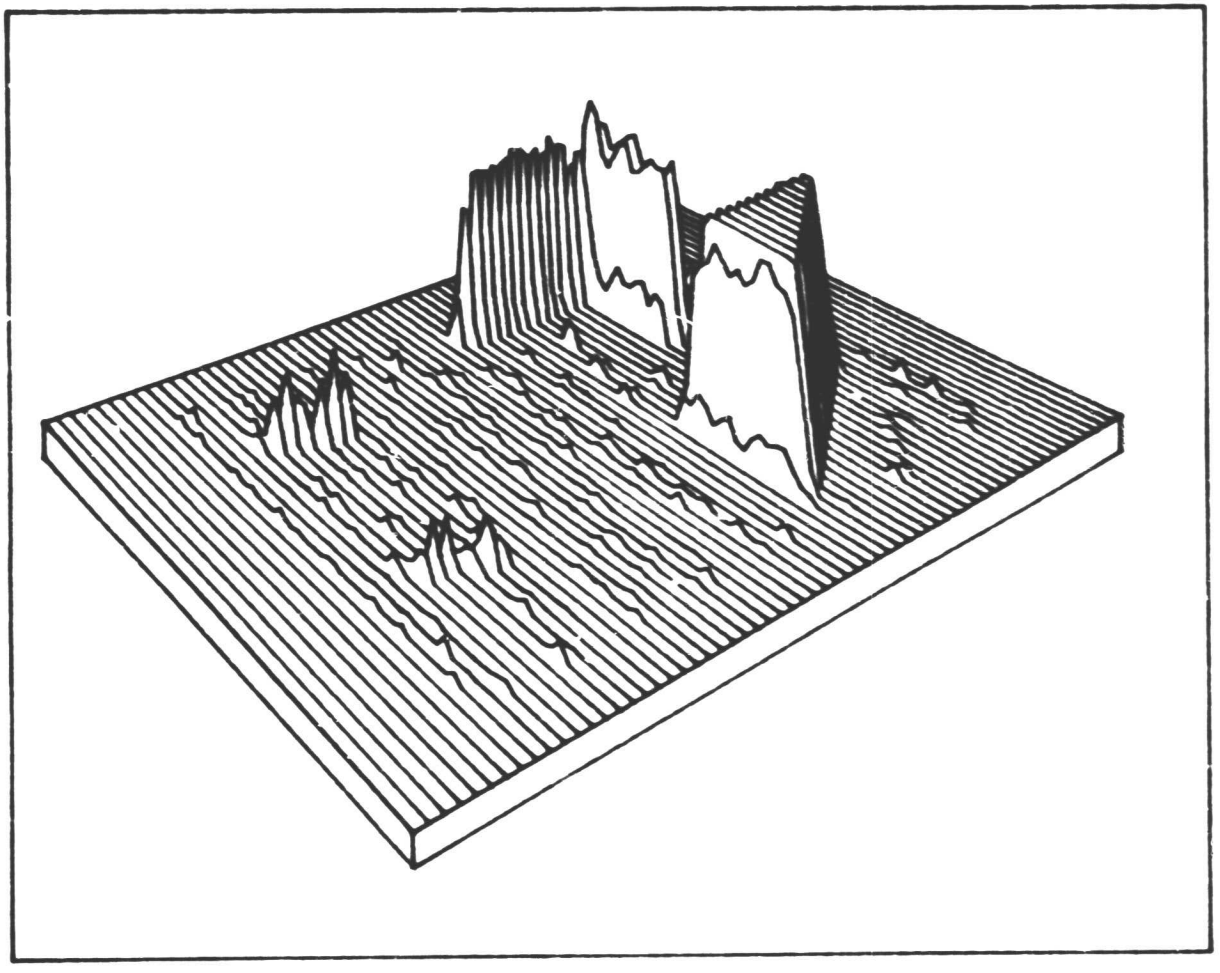
UNFOLDING ITERATION 10X2 - 20 SMOOTHINGS - SNR=200

Figure 4.45



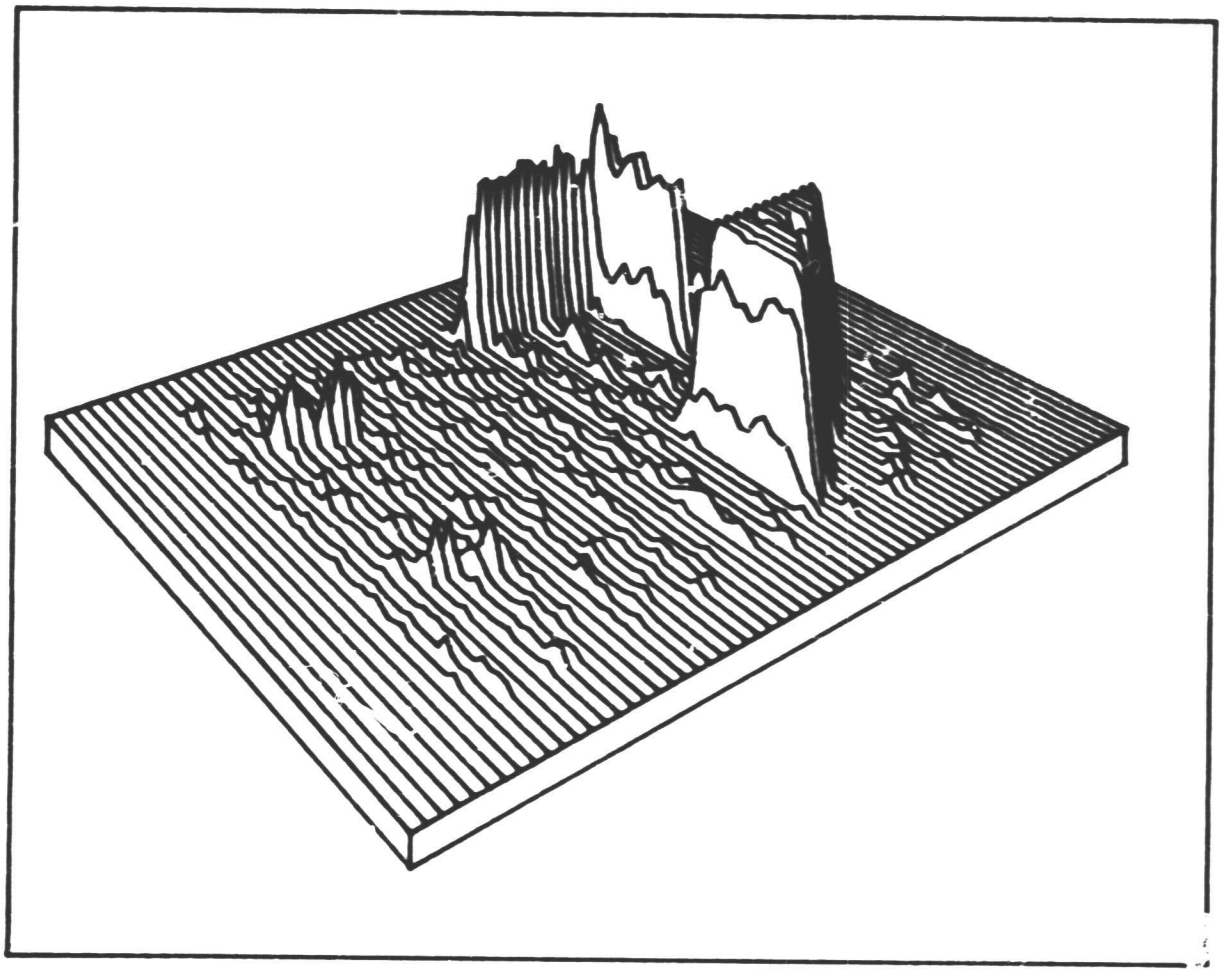
UNFOLDING ITERATION 20X2 - 20 SMOOTHINGS - SNR=200

Figure 4.46



UNFOLDING ITERATION 50X2 - ALL CONSTRAINTS - SNR=200

Figure 4.47



UNFOLDING ITERATION 4X30 - 20 SMOOTHINGS - SNR=200

Figure 4.48

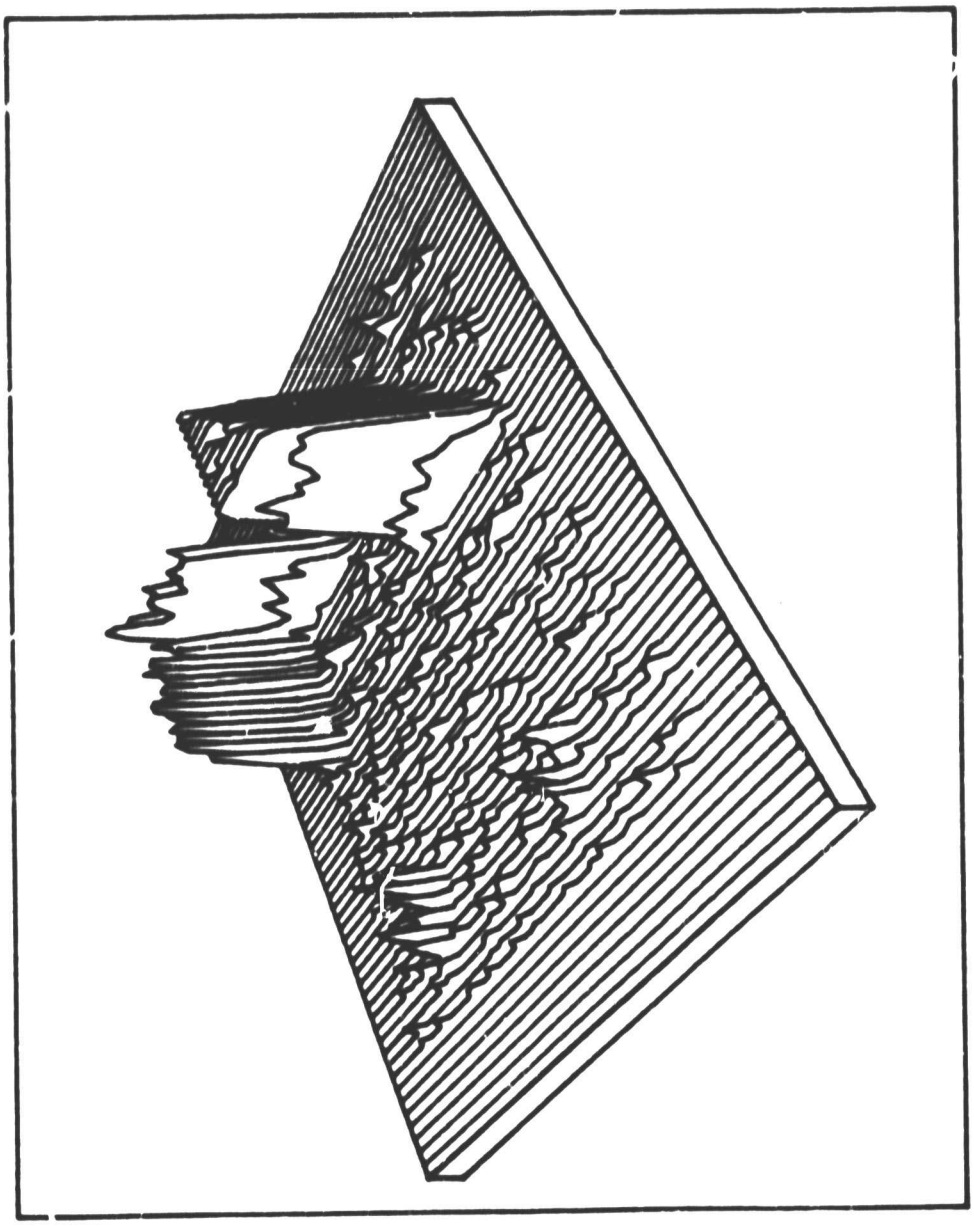
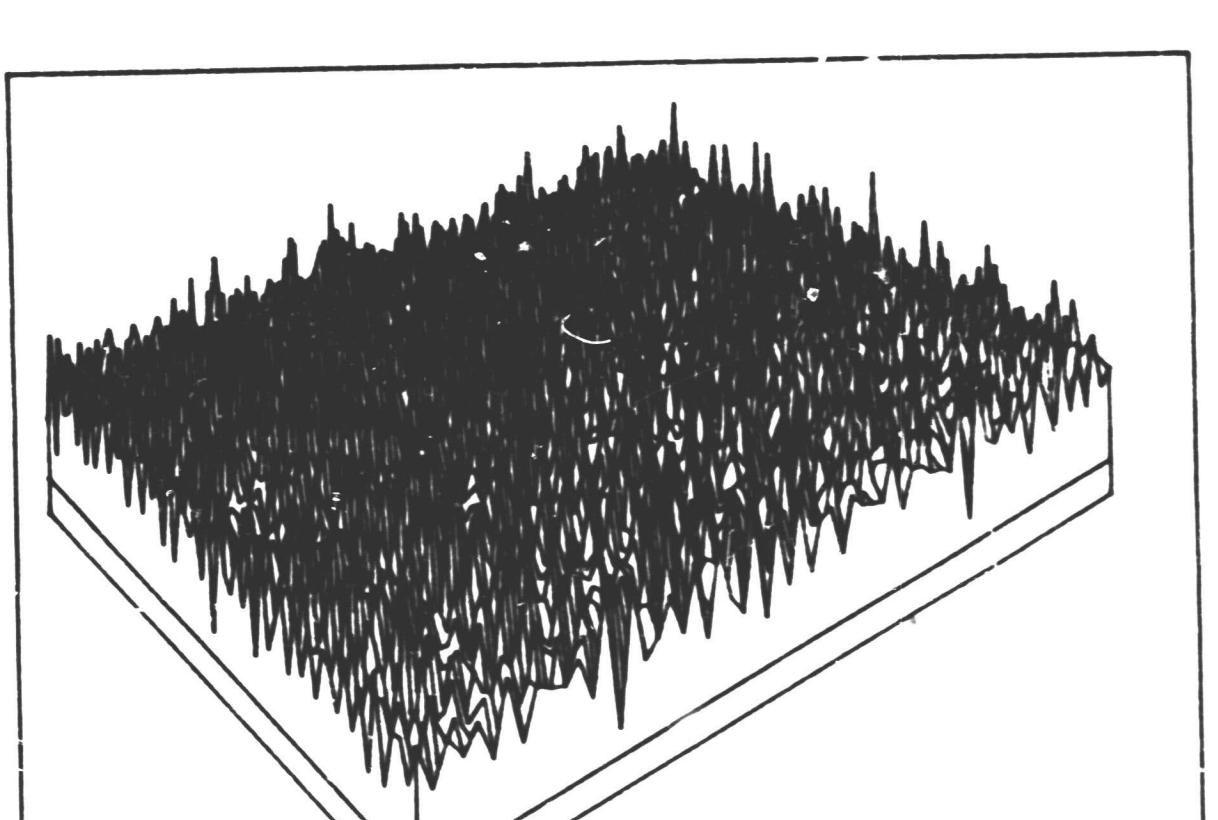
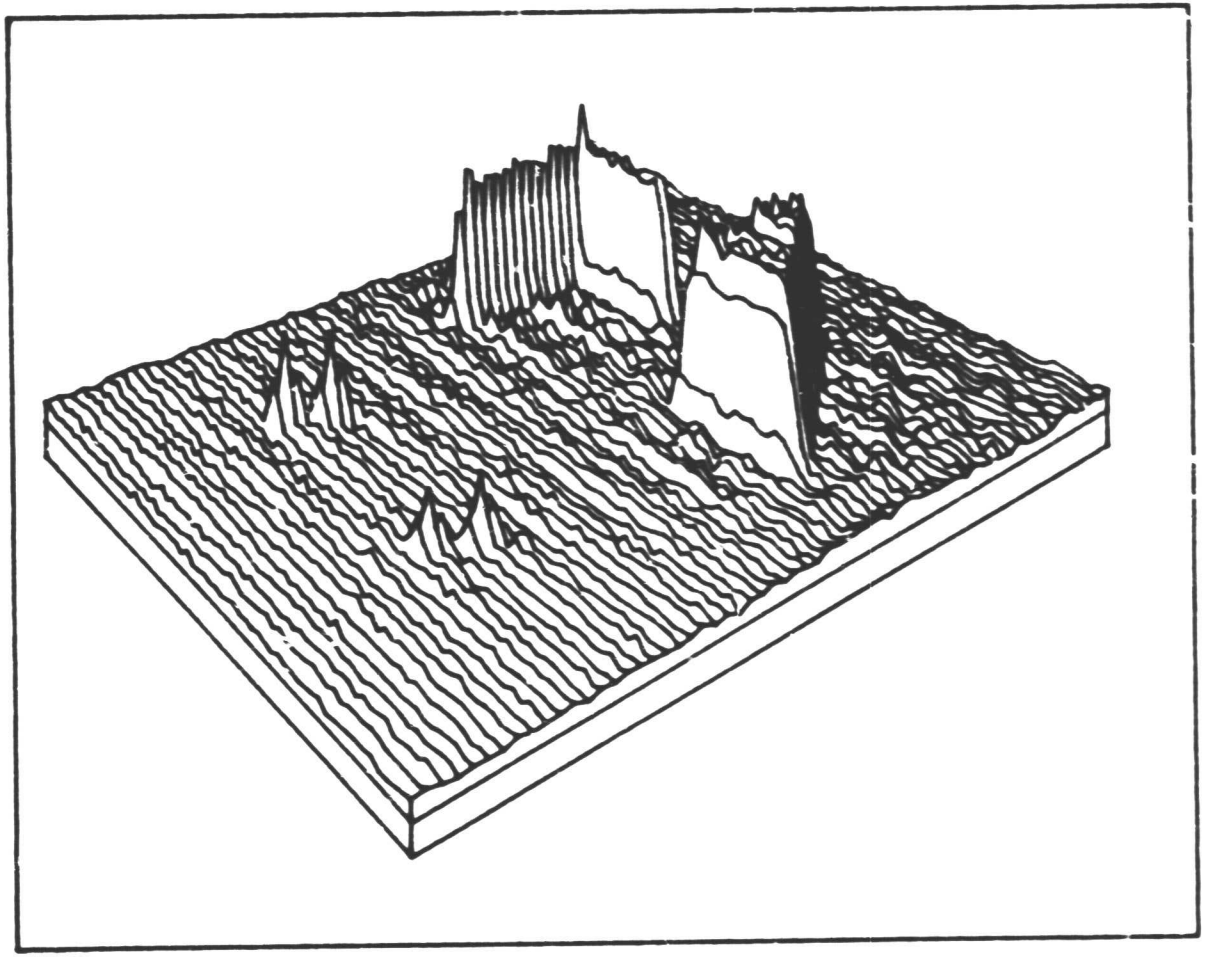


Figure 4.49



ρ_p - NO SMOOTHING - SNR=2000

Figure 4.50



ρ_ρ - 60 SMOOTHINGS - SNR=2000

Figure 4.51

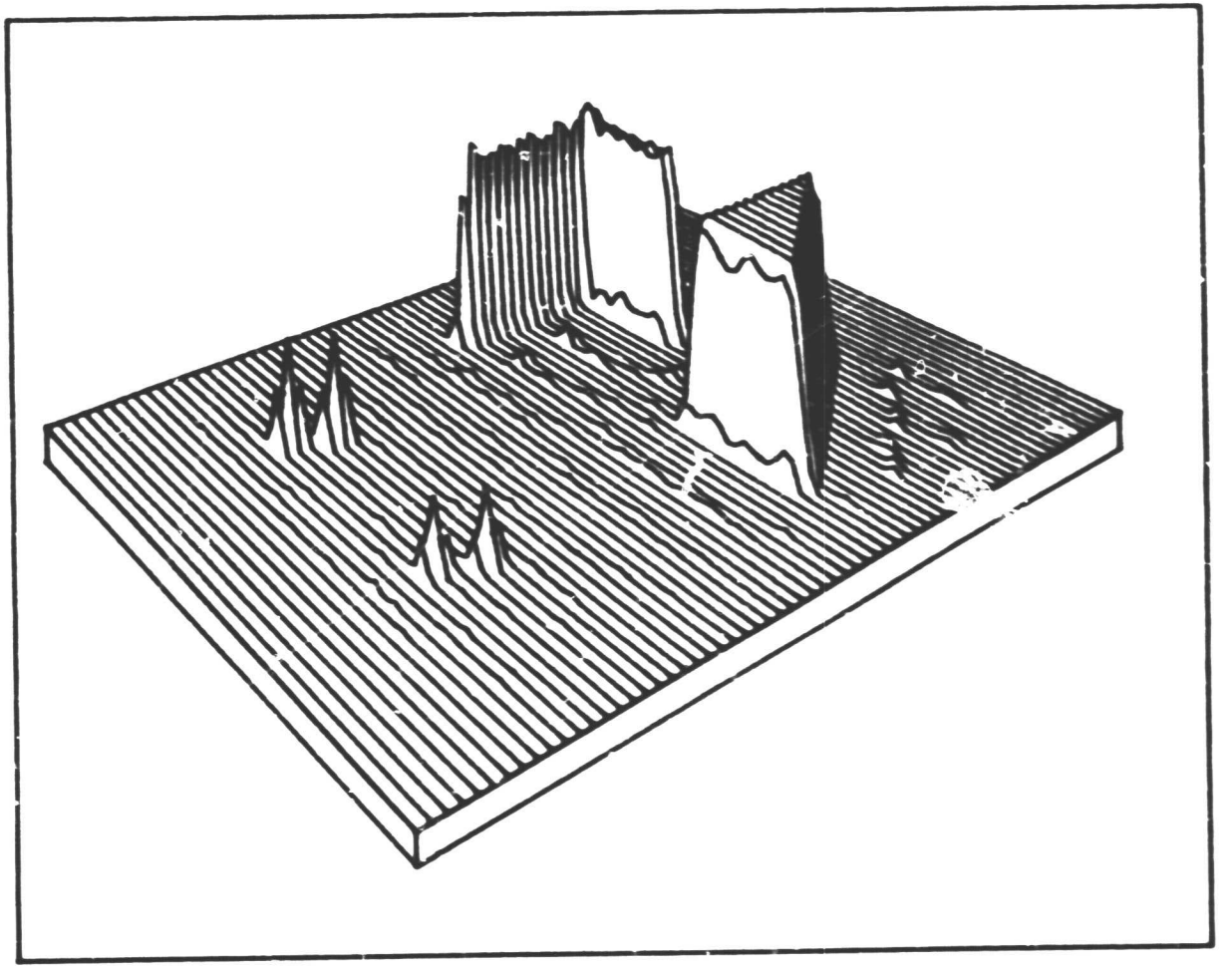


Figure 4.52

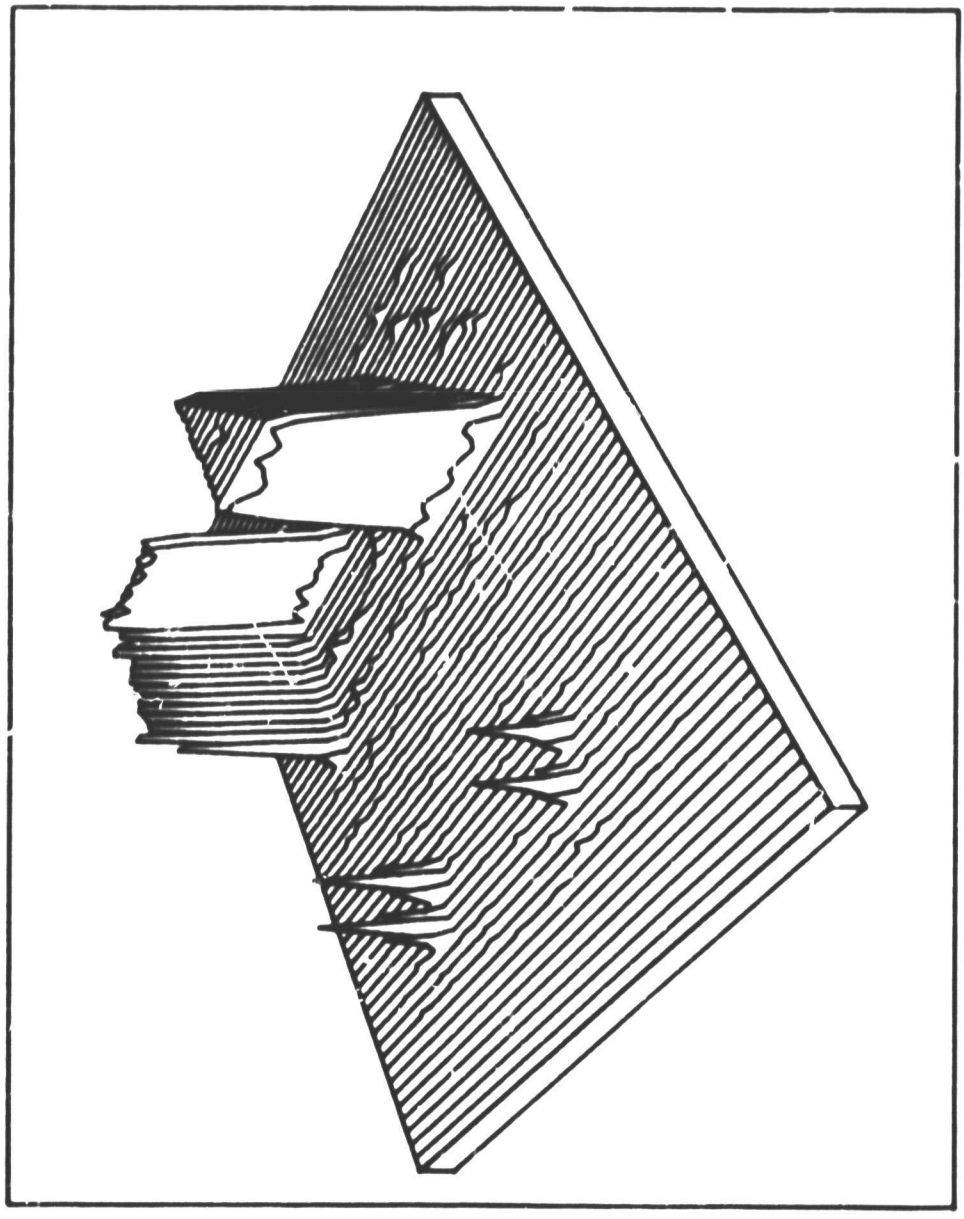


Figure 4.53

APPENDIX ONE

The main routine, F5DCON.FOR, used to implement the always-convergent iterative techniques for noise removal and deconvolution is described in this appendix. Input to the program must include a two-dimensional array of numbers representing the system response function h(x,y) and another array of sample values corresponding to the system output g(x,y). The program is written in such a way that modifying parameters DIM1 and DIM2 to reflect the dimensions of the desired working arrays is all that is necessary to change the value referred to as N in chapters two and three. DIM1 must be assigned a value of N and DIM2 the value 2N before the program is compiled. This program makes use of complex data arrays and FORTRAN supplied complex arithmetic functions in order to simplify the handling of complex functions. Declared as complex quantities are two-dimensional arrays G(I,J), H(I,J), and Hs(I,J), each having dimension N by N. Declared as real arrays of dimension 2N by N and equivalenced to the complex arrays are the arrays Gin(I,J), Hin(I,J), and Hout(I,J). The last three arrays are used for convenience in doing I/O of the real parts of the complex arrays and in applying constraints. A complex array of dimensions N by N is treated by FORTRAN as an array of real pairs representing (real, imaginary) values of a single complex variable. The equivalencing done above simply makes the real and imaginary

parts of each complex value accessible by subscripting instead of by a call to a library function. All multi-dimensional arrays are stored in core by FORTRAN in such a way that the first index varies fastest. This is diagrammed in the comments within the main routine.

Input to the program is mainly interactive. Operator input is requested to supply the filenames for g(x,y) and h(x,y) and the dimensions of g(x,y) and h(x,y). These last dimensions are independent of N, which must be chosen such that $n_g+n_h-1<N$. Here, the values of n_g and n_h represent the largest dimension of g(x,y) and h(x,y), and the program expects to read in square arrays for g(x,y) and h(x,y). Also requested by the program is the FORTRAN format in which the input data are organized.

The remaining input to the program determines the processing parameters to be used. Requested are the total number of iterations of smoothing and unfolding to be performed, the interval at which constraints are to be applied and the output interval for the unfolding iteration. Also requested is the name of the file containing the data to be unfolded, and an initial numeric offset value to be used in naming the files output by the unfolding iteration. Files output by the smoothing routine are named SMNNNN.DAT where NNNN is a number representing the number of equivalent time domain iterations performed on g(x,y) to generate each one. Output files from the unfolding routine are named in

the same manner, but are written only if the number of constraint applications performed for the current iteration is zero modulo the output iteration interval. The program may be restarted in order to continue unfolding a previous result by providing the appropriate filename offset and replacing g(x,y) with the original h(x,y) and replacing h(x,y) with the result from the last run. Output file names are UFNNNN.DAT where NNNN = filename offset + (constraint application interval)X(number of constraint applications). This number represents the total number of equivalent time domain iterations used in generating the result data.

The last control data input to the program specifies the way in which constraints will be applied. The operator must enter levels for application of the lower limit and upper limit constraints. To specify the finite extent constraint, the operator enters values corresponding to the upper left and lower right corners of the non-zero region of the image. Since the program always applies these constraints at the end of each iteration, to turn one off requires entering a value which makes that particular constraint ineffective.

Subroutine DDFFT, a modified version of the two-dimensional FFT routine written by Kathleen Whitehorn [Whitehorn, K (1980)], is used by the main routine to perform most operations. Modifications include the use of complex arrays and complex arithmetic, and reduction of the

number of arguments to one complex array and two integers. Also, to increase the efficiency of internally passing data to the one-dimensional FFT (subroutine "FFT") within the subroutine, the array passed to this routine is actually just one row of the complex two-dimensional array used by the main routine. This eliminates a large amount of data shuffling which would be required to fill another input array on every call to FFT. Large increases in efficiency would result by converting to an FFT which makes use of the symmetries of the hermitian transform corresponding to real input data. Recognizing these symmetries allows the array dimensions to be cut in half, and reduces the amount of computation required by the same amount.

NOTE: - All programs in the listings use a slightly different notation from that used in the body of this thesis. In the programs and the appendices, g(x,y) is used to represent the PSF of the system and h(x,y) represents the output of the system in response to the input f(x,y). This amounts to a reversal of the roles of g(x,y) and h(x,y) in the version of the notation used in the programs.

PROGRAM P5DCON.FOR

```
C********** GFORM and HFORM are variable format spec's
C********** FILNAM is a variable file name
      DOUBLE PRECISION GFORM(5),HFORM(5),FILNAM
      DOUBLE PRECISION FILIN,GFILE,HFILE
      INTEGER UR,LR,LC,UC,MULT
      REAL INTSM,INTUF
C********** A complex array is stored with the first
C********** index varying fastest.  Equivalenced to
C********** each array is a corresponding real array
C********** for convenience in doing I/O.
C********** Data arrays are assumed to be arranged:
C**********  .....I.....
C**********  .      row 1      .
C**********  .      row 2      .
C**********  .        .        .
C**********  J        .        .
C**********  .        .        .
C**********  .      row N-1    .
C**********  .      row N      .
C**********  .....I.....
      COMPLEX G(64,64),H(64,64),Hs(64,64)
      DIMENSION Gin(128,64),Hin(128,64),Hout(128,64)
      EQUIVALENCE (G(1,1),Gin(1,1)),(H(1,1),Hin(1,1))
      EQUIVALENCE (Hs(1,1),Hout(1,1))
C********** Array Gm is for storage of the magnitude of G
      REAL Gm(64,64)
      TYPE 16
```

```
16      FORMAT(' Is this a restart? (0=NO) ',$)
        ACCEPT 2,IRSTRT
        TYPE 1
1       FORMAT(' Enter g filename ',$)
        ACCEPT 18,GFILE
2       FORMAT(2I)
        TYPE 17
17      FORMAT(' Enter h filename ',$)
        ACCEPT 18,HFILE
18      FORMAT(A10)
        TYPE 3
3       FORMAT(' Enter g and h dimensions (2I) ',$)
        ACCEPT 2,NG,NH
        TYPE 6
6       FORMAT(' Enter g format (A50) ',$)
        ACCEPT 7,(GFORM(I),I=1,5)
        TYPE 8
8       FORMAT(' Enter h format (A50) ',$)
        ACCEPT 7,(HFORM(I),I=1,5)
7       FORMAT(5A10)
        TYPE 4
4       FORMAT(' Enter   smoothings,   unfoldings ',$)
        ACCEPT 2,NHS,NUNF
        TYPE 9
9       FORMAT(' Enter unfolding computation interval ',$)
        ACCEPT 507,INTUF
        TYPE 15
```

```
15      FORMAT(' Enter output iteration interval ',$)
        ACCEPT 2,MULT
        TYPE 10
10      FORMAT(' Enter smoothing computation interval ',$)
        ACCEPT 507,INTSM
        TYPE 5
5       FORMAT(' Enter filename offset ',$)
        ACCEPT 2,INIT
        TYPE 19
19      FORMAT(' Enter filename for unfolding ',$)
        ACCEPT 18,FILIN
        TYPE 11
11      FORMAT(' Enter black, white level constraints ',$)
        ACCEPT 507,BLACK,PEAK
C********** Finite extent limits are entered as two pairs:
C********** first pair is lower numbered row,column (upper
C********** left corner) second pair is lower right corner
        TYPE 12
12      FORMAT(' Enter finite extent constraint limits ',$)
        ACCEPT 13,LR,LC,UR,UC
13      FORMAT(4I)
C********** Set firm parameters
        ISIZ=64
        TOL=.000001
507     FORMAT(64G)
C********** Zero input arrays
        DO 105 I=1,ISIZ
```

```
          DO 105 J=1,ISIZ
          G(I,J)=0.
105       H(I,J)=0.
C********** Input data
          OPEN (UNIT=32,ACCESS='SEQIN',FILE=GFILE)
          READ (32,GFORM) ((Gin(I,J),I=1,NG*2,2),J=1,NG)
C********** Compute G(u,v) — ISIZ X ISIZ -i transform
          ISIGN=-1
          CALL DDFFT(ISIZ,ISIGN,G)
C********** Compute Gm
          DO 104 I=1,ISIZ
          DO 104 J=1,ISIZ
104       Gm(I,J)=CABS(G(I,J))
C********** Normalize Gm
          GmMAX=1./Gm(ISIZ/2+1,ISIZ/2+1)

          DO 101 I=1,ISIZ
          DO 101 J=1,ISIZ
101       Gm(I,J)=Gm(I,J)*GmMAX
          WRITE (35,507) ((Gm(I,J),I=1,ISIZ),J=1,ISIZ)
C********** Compute H(u,v)
          OPEN (UNIT=32,ACCESS='SEQIN',FILE=HFILE)
          READ (32,HFORM) ((Hin(I,J),I=1,NH*2,2),J=1,NH)
          IF (IRSTRT.NE.0) GOTO 321
C********** Normalize h to f
          DO 311 I=1,ISIZ
          DO 311 J=1,ISIZ
```

```
311       H(I,J)=H(I,J)*GmMAX
C********** Save normalized original h as SM0.DAT
321       OPEN  (UNIT=30,ACCESS='SEQOUT',FILE='SM0.DAT')
          WRITE (30,507) ((Hin(I,J),I=1,ISIZ*2,2),J=1,ISIZ)
          CLOSE (UNIT=30,ACCESS='SEQOUT',FILE='SM0.DAT')
C********** Compute H from h
          ISIGN=-1
          CALL DDFFT(ISIZ,ISIGN,H)
          IF ( NHS .EQ. 0 ) GOTO 400
C********** Smoothing
C********** Compute Hn = H * [1-(1-Gm)**n]
          DO 107 K=1,NHS
          DO 103 I=1,ISIZ
          DO 103 J=1,ISIZ
103       Hs(I,J)=H(I,J) * ( 1- ( 1- Gm(I,J) )**(INTSM*K) )
C********** Transform back to get hs
          ISIGN=1
          CALL DDFFT(ISIZ,ISIGN,Hs)
C********** Output smoothed h
          KINT=K*INTSM
          ENCODE(10,501,FILNAM) KINT
501       FORMAT('SM',I4,'.DAT')
          TYPE 502,FILNAM
502       FORMAT(1X,A10)
          OPEN  (UNIT=30,ACCESS='SEQOUT',FILE=FILNAM)
503       WRITE (30,507) ((HOUT(I,J),I=1,ISIZ*2,2),J=1,ISIZ)
          CLOSE (UNIT=30,ACCESS='SEQOUT',FILE=FILNAM)
```

```
107     CONTINUE
C********** UNFOLDING
C********** read input to unfolding routine
400     TYPE 403,FILIN
403     FORMAT(' UNFOLDING FILE:  ',A10)
        OPEN (UNIT=32,ACCESS='SEQIN',FILE=FILIN)
C********** Zero Im part of Hs array
        DO 404 I=2,2*ISIZ,2
        DO 404 J=1,ISIZ
404     Hout(I,J)=0.
        READ (32,507) ((Hout(I,J),I=1,ISIZ*2,2),J=1,ISIZ)
C********** Compute Hs from hs input
        ISIGN=-1
        CALL DDFFT(ISIZ,ISIGN,Hs)
C********** Compute Fp = H/G
C********** Get (G*)/(Gm)**2 = 1/G first
401     DO 305 I=1,ISIZ
        DO 305 J=1,ISIZ
        IF ( Gm(I,J) .LT. TOL ) GOTO 303
        TMP=GmMAX/(Gm(I,J))**2
        GOTO 305
303     TMP=0
305     G(I,J)=CONJG(G(I,J))*TMP
        WRITE(34,508)((G(I,J),I=1,ISIZ),J=1,ISIZ)
508     FORMAT(128G)
        IF (IRSTRT.EQ.0) GOTO 318
C********** Do H * (1/G) if this is a restart
```

```
          DO 306 I=1,ISIZ
          DO 306 J=1,ISIZ
306       H(I,J)=H(I,J)*G(I,J)
          GOTO 320
C********** Do Hs * (1/G) if this is not a restart
318       DO 319 I=1,ISIZ
          DO 319 J=1,ISIZ
319       H(I,J)=Hs(I,J)*G(I,J)
320       WRITE(33,508)((H(I,J),I=1,ISIZ),J=1,ISIZ)
C********** H IS NOW Fp=(H/G) (PRINCIPAL SOLUTION)
C********** Compute (1-Gm)**INTUF
          DO 309 I=1,ISIZ
          DO 309 J=1,ISIZ
309       Gm(I,J)=(1-Gm(I,J))**INTUF
C********** Now ready to iterate
          DO 301 K=1,NUNF
C********** NOW GET Hs = Fp - (Fp - Hs)*(1-Gm)**INTUF
          DO 307 I=1,ISIZ
          DO 307 J=1,ISIZ
307       Hs(I,J)=H(I,J) - ( H(I,J) - Hs(I,J) ) * Gm(I,J)
C********** Compute time domain result hs
          ISIGN=1
          CALL DDFFT(ISIZ,ISIGN,Hs)
C********** Apply time domain constraints
          DO 308 I=1,2*ISIZ,2
          DO 308 J=1,ISIZ
          IF ( Hout(I,J) .GT. PEAK ) Hout(I,J)=PEAK
```

```
308       IF ( Hout(I,J) .LT. BLACK ) Hou.(I,J)=BLACK
C********** Apply finite extent constraint
          IF ( UR .EQ. 0 ) GOTO 316
          DO 312 I=1,2*LC,2
          DO 312 J=1,ISIZ
312       HOUT(I,J)=0.
          DO 313 I=2*UC+1,2*ISIZ,2
          DO 313 J=1,ISIZ
313       HOUT(I,J)=0.
          DO 314 I=2*LC+1,2*UC,2
          DO 314 J=1,LR
314       HOUT(I,J)=0.
          DO 315 I=2*LC+1,2*UC,2
          DO 315 J=UR,ISIZ
315       HOUT(I,J)=0.
C********** Clear imaginary garbage
316       DO 310 I=2,2*ISIZ,2
          DO 310 J=1,ISIZ
310       HOUT(I,J)=0.
          IF (MOD(K,MULT) .NE. 0) GOTO 317
C********** Output hs
          KINT=K*INTUF+INIT
          ENCODE(10,505,FILNAM) KINT
505       FORMAT('UF',I4,'.DAT')
          TYPE 502,FILNAM
          OPEN  (UNIT=31,ACCESS='SEQOUT',FILE=FILNAM)
506       WRITE (31,507) ((Hout(I,J),I=1,ISIZ*2,2),J=1,ISIZ)
```

```
      CLOSE (UNIT=31,ACCESS='SEQOUT',FILE=FILNAM)
C********** If K=NUNF quit
317   IF( K .EQ. NUNF) GOTO 301
C********** Transform back
      ISIGN=-1
      CALL DDFFT(ISIZ,ISIGN,Hs)
301   CONTINUE
      END
```

APPENDIX TWO

Program FILTER.FOR is a general purpose routine which uses various subroutines to accomplish time domain convolution and Fourier transforms. It is the routine used to perform most of the operations required to generate synthetic data from the model of Chapter One. Input to the routine is specified by interactive dialogue as described in Appendix One for the program F5DCON.FOR. Function number one calls the subroutine CON2X.FOR, which performs an expanding time domain convolution. This implies that the output array will be larger than either of the two input arrays, and the resultant dimensions are output to the operator at runtime. The second function executes the non-expanding convolution subroutine, which queries for the subscripts defining the origin of g(x,y). This origin determines which portions of the expanded output array will not be computed, and the result will have the same dimensions as the input array f(x,y). Execution of function three results in a call to a version of DDFFT.FOR which performs a FFT on f(x,y) and returns F(u,v) in the output array H(I,J).

Following the listings of the above main program and subroutines is a listing of program NOISE.FOR which performs the addition of gaussian noise to a two-dimensional input array. The number (I) which it requests at start-up is used to generate (I) calls to the FORTRAN pseudorandom number

generator RAN in order to allow for the production of different sets of noise data on different runs. If I is the same for any two runs, then the generated noise samples for those two runs will also be the same. Output by the routine are two data files, one containing the input data plus noise, and the other containing the noise alone. The numbers generated by the function RAN have a mean of .5 and a range of zero to one. Production of Gaussian noise from a sequence of uniformly distributed random numbers is performed by subroutine GAUSS.FOR. By the central-limit theorem, the probability density distribution for a sum of uniform random numbers approaches the normal distribution as the number of terms in the sum grows large. Since the variance of a uniform distribution of random numbers between zero and one is 1/12, GAUSS.FOR sums 12 random numbers to achieve a nearly gaussian distribution with a variance of one in the sum [Hamming (1962)]. It then adjusts the mean of the sum back to zero and multiplies the result by the desired variance to compute the noise sample.

The last listing is an example of a time-domain implementation of the always-convergent techniques. The $g(x,y)$ input to the program must be $h_m(x,y)$ as defined in Chapter Two. This routine expects input files to be unformatted, random access binary data dimensioned 64 by 64. Output is in the same format. Here the record number serves to indicate the iteration number, since no iterations may be skipped.

```
                PROGRAM FILTER.FOR
      DIMENSION F(100,100),G(100,100),H(100,100),FFORM(10),GFORM(10)
      DOUBLE PRECISION FFILE,GFILE
      INTEGER FX,F ,GX,GY,HX,HY
      COMMON F,G,FX,FY,GX,GY,H,HX,HY,ERR,FFORM,GFORM
      DATA FFORM(1)/1H(/,FFORM(10)/1H)/,GFORM(1)/1H(/,GFORM(10)/1H)/
      MAINX=100
      MAINY=100
11    TYPE 1
1     FORMAT( ' ENTER F DIMENSIONS, G DIMENSIONS ',$)
      ACCEPT 2,FX,FY,GX,GY
2     FORMAT(4I)
      TYPE 3
3     FORMAT( ' ENTER F FILENAME ',$)
      ACCEPT 4,FFILE
4     FORMAT(A10)
      TYPE 5
5     FORMAT( ' ENTER G FILENAME ',$)
      ACCEPT 4,GFILE
16    TYPE 6
6     FORMAT( ' ENTER F FORMAT ',$)
      ACCEPT 7,(FFORM(I),I=2,9)
7     FORMAT(10A5)
18    TYPE 8
8     FORMAT( ' ENTER G FORMAT ',$)
      ACCEPT 7,(GFORM(I),I=2,9)
C
```

```
C       NOW WE HAVE THE INFO NECESSARY TO DIMENSION F_G AND
C       READ THE INPUT DATA...
C
C
C       READ F AND G
C
        OPEN (UNIT=20,ACCESS='SEQIN',FILE=FFILE)
        OPEN (UNIT=21,ACCESS='SEQIN',FILE=GFILE)
        READ (20,FFORM,END=21) ((F(I,J),I=1,FX),J=1,FY)
        READ (21,GFORM,END=22) ((G(I,J),I=1,GX),J=1,GY)
        GOTO 20
21      TYPE 27
27      FORMAT( ' RAN OVER END OF FILE ON F ')
        ERR=-1
        GOTO 16
22      TYPE 28
28      FORMAT( ' RAN OVER END OF FILE ON G ')
        ERR=-1
        GOTO 18
20      CALL CRUNCH
        IF (ERR.NE.0) GOTO 11
C       SUBROUTINE FUNCTION EXECUTES A SELECTED SUBROUTINE
C       AND COMMUNICATES VIA BLANK COMMON
        CALL FINISH
C       SUBROUTINE FINISH PROVIDES GENERAL I/O FOR COMMON ARRAYS
        END
C****   END OF MAIN PROGRAM   ****C
```

```
      SUBROUTINE CRUNCH
      DIMENSION F(100,100),G(100,100),H(100,100)
      INTEGER FX,FY,GX,GY,ERR
      COMMON  F,G,FX,FY,GX,GY,H,HX,HY,ERR
C
C     NOW THE DATA IS IN CORE, SO DO SOMETHING TO IT
C
6     TYPE 3
3     FORMAT( ' ENTER FUNCTION     ',$)
      ACCEPT 4,K
4     FORMAT(I)
      GOTO (10,11,20) K
C
C     HERE IMPLIES INVALID K
C
      TYPE 5
5     FORMAT( ' TRY A VALID FUNCTION ')
      GOTO 6
10    CALL CON2X(F,G,FX,FY,100,100,GX,GY,100,100,
     *           H,HX,HY,100,100,ERR)
      IF (ERR.NE.0) GOTO 6
      RETURN
11    TYPE 12
12    FORMAT(' ENTER PEAKX,PEAKY  ',$)
      ACCEPT 13,PEAKX,PEAKY
13    FORMAT(2I)
      CALL CON2(F,G,FX,FY,100,100,GX,GY,100,100,
```

```
     *            H,HX,HY,100,100,PEAKX,PEAKY,ERR)
          IF (ERR.NE.0) GOTO 6
          RETURN
20        CALL DDFFT(F,G,FX,FY,100,100,GX,GY,100,100,
     *            H,HX,HY,100,100,ERR)
          RETURN
          END
          SUBROUTINE FINISH
          DIMENSION F(100,100),G(100,100),H(100,100)
          INTEGER FX,FY,GX,GY,HX,HY,ERR
          COMMON F,G,FX,FY,GX,GY,H,HX,HY,ERR,FFORM,GFORM
          DIMENSION FFORM(10),GFORM(10)
          DOUBLE PRECISION OFILE,FFILE,GFILE
          DIMENSION OFORM(10)
          DATA OFORM(1)/1H(/,OFORM(10)/1H)/
          TYPE 10
10        FORMAT(' DO YOU WANT F AND G OUTPUT? ',$)
          ACCEPT 11,ANS
11        FORMAT(A5)
          IF (ANS.NE.'YES') GOTO 20
          TYPE 12
12        FORMAT(' ENTER F,G FILENAMES ON 2 SEPARATE LINES ',$)
          ACCEPT 3,FFILE
          ACCEPT 3,GFILE
          OPEN (UNIT=30,FILE=FFILE,ACCESS='SEQOUT')
          OPEN (UNIT=31,FILE=GFILE,ACCESS='SEQOUT')
          WRITE(30,FFORM)((F(I,J),I=1,FX),J=1,FY)
```

```
      WRITE(31,GFORM)((G(I,J),I=1,GX),J=1,GY)
20    TYPE 1
1     FORMAT(' ENTER H OUTPUT FILENAME ',$)
      ACCEPT 3,OFILE
3     FORMAT(A10)
      TYPE 2,HX,HY
2     FORMAT(' HX=',I4,' HY=',I4,'   ENTER FORMAT SPEC ',$)
      ACCEPT 4,(OFORM(I),I=2,9)
4     FORMAT(10A5)
      OPEN(UNIT=22,FILE=OFILE,ACCESS='SEQOUT')
      WRITE(22,OFORM)((H(I,J),I=1,HX),J=1,HY)
      RETURN
      END
```

```
                    PROGRAM NOISE.FOR
          DIMENSION A(64,64),Y(64)
          TYPE 31
31        FORMAT(' ENTER A NUMBER (I)',$)
          ACCEPT 2,IR
          TYPE 1
1         FORMAT(' Input file   ',$)
          ACCEPT 2,IF
          TYPE 3
3         FORMAT(' Output file   ',$)
          ACCEPT 2,IO
2         FORMAT(I)
          TYPE 4
4         FORMAT(' Alpha = ',$)
          ACCEPT 5,ALPHA
5         FORMAT(G)
          READ (IF,6,END=7)((A(I,J),I=1,64),J=1,64)
6         FORMAT(64G)
          OPEN(UNIT=60,ACCESS='SEQOUT',FILE='NOISE.DAT')
C******_GENERATE GAUSSIAN NOISE
          RMS=0.
          AVG=0.
          GOTO 13
7         TYPE 12
12        FORMAT(' SHORT INPUT FILE?')
13        DO 30 I=1,IR
30        P=RAN(1)
```

```
      DO 9 J=1,64
      DO 8 I=1,64
      CALL GAUSS(ALPHA,0,Y(I))
      RMS=RMS+Y(I)**2
      AVG=AVG+Y(I)
      A(I,J)=A(I,J)+Y(I)
8     IF (A(I,J) .LT. 0.) A(I,J)=0.
9     WRITE(60,10)(Y(K),K=1,64)
      RMS=(RMS/4096.)**.5
      AVG=AVG/4096.
      TYPE 14,RMS
      TYPE 14,AVG
14    FORMAT(1X,G)
      WRITE(IO,11)((A(I,J),I=1,64),J=1,64)
10    FORMAT(64F)
11    FORMAT(64F8.3)
      END
```

PROGRAM GAUSS.FOR

```
C       Subroutine GAUSS
C
C       PURPOSE
C
C               Computes a normally distributed random number with
C               a given mean and standard deviation
C
C       USAGE
C
C               CALL GAUSS(S,AM,V)
C
C
C       DESCRIPTION OF PARAMETERS
C
C               S - the desired standard deviation of the normal
C                       distribution
C               AM - the desired mean of the normal distribution
C               V - the value of the computed normal random variable
C
C       REMARKS
C
C               This subroutine uses a machine specific uniform
C               random number generator
C
C       METHOD
C
```

```
C               Uses 12 uniform random numbers to compute normal
C               random numbers by central limit theorem.  The result
C               is then adjusted to match the given mean and standard
C               deviation.  The uniform random numbers computed within
C               the subroutine are computed by the FORTRAN "RAN" function.
C
C
C
      SUBROUTINE GAUSS(S,AM,V)
      A=0.0
      DO 1 I=1,12
1     A=A+RAN(1)
      V=(A-6.0)*S+AM
      RETURN
      END
```

```
                PROGRAM DCONTU.FOR
      INTEGER PEAK,OUTX,OUTY
      DIMENSION G(64,64),H(64,64),HN(64,64),HNM1(64,64)
      TYPE 1
1     FORMAT(' ENTER G AND H UNIT NUMBERS')
      ACCEPT 2,IG,IH
2     FORMAT(2I)
      TYPE 3
3     FORMAT(' ENTER G AND H DIMENSIONS (2I)')
      ACCEPT 2,NG,NH
      TYPE 4
4     FORMAT(' ENTER   SMOOTHINGS,   UNFOLDINGS')
      ACCEPT 2,NHS,NUNF
      TYPE 5
5     FORMAT(' ENTER RECORD   FOR UNFOLDING')
      ACCEPT 2,IREC
      CALL DEFINE FILE (30,4096,LOC1,0,0,0)
      CALL DEFINE FILE (31,4096,LOC2,0,0,0)
C
C******_INPUT DATA
C
      READ (IG,10) ((G(I,J),I=1,NG),J=1,NG)
      READ (IH,11) ((H(I,J),I=1,NH),J=1,NH)
10    FORMAT(32G)
11    FORMAT(32G)
C
C******_NORMALIZE G
```

```
C
      SUM=0
      DO 101 I=1,NG
      DO 101 J=1,NG
101   SUM=SUM+G(I,J)
      SUMINV=1./SUM
      DO 102 I=1,NG
      DO 102 J=1,NG
102   G(I,J)=G(I,J)*SUMINV
C
C******_ZERO HNM1 ARRAY
C
      DO 103 I=1,NH
      DO 103 J=1,NH
103   HNM1(I,J)=0.
C
      IF(NS.EQ.0) GOTO 700
C
C
C******_1'st MORRISON ITERATION
C
100   CALL CON2(H,G,32,32,64,64,32,32,64,64,HNM1,
     "          OUTX,OUTY,64,64,17,17,ERR)
C
C
C******_WRITE SMOOTHED H
C
```

```
      WRITE (30 1) HNM1
C
C
C******_2'nd MORRISON ITERATION
C
      DO 200 I10=2,NHS
C
      ERR=0
C
C******_H(N)=H - H(N-1)
C
      DO 201 I=1,NH
      DO 201 J=1,NH
201   HN(I,J)=H(I,J)-HNM1(I,J)
C
C******_H(N) = H(N-1) + [ H - H(N-1) ] * G
C
202   CALL CON2(HN,G,32,32,64,64,32,32,64,64,HNM1,
     "          OUTX,OUTY,64,64,17,17,ERR)
C
C
C******_COMPUTE RMS DIFFERENCE BETWEEN LAST 2 ITERATIONS
C
      NH2=NH*NH
C
      DO 203 I=1,NH
      DO 203 J=1,NH
```

```
203       ERR=ERR+(HNM1(I,J)+HN(I,J)-H(I,J))**2
C
          ERR=(ERR/NH2)**.5
C
C******_OUTPUT ITERATION
C
          WRITE (30 I10) HNM1
200       WRITE(20,6)I10,ERR
6         FORMAT(X,'ITERATION   ',I,5X,'RMS CHANGE   ',G)
C
C******_UNFOLDING
C
C
700       READ (30 IREC) HNM1
C
C
          DO 300 I11=1,NUNF
C
C
          DO 305 I=1,NH
          DO 305 J=1,NH
305       HN(I,J)=H(I,J)-HNM1(I,J)
C
C******_H(N) = (FP - HNM1)
C
301       CALL CON2(HN,G,32,32,64,64,32,32,64,64,HNM1,
     "              OUTX,OUTY,64,64,17,17,ERR)
```

```
C
C
C******_H(N) = H(N-1) + [ FP - H(N-1) ] * G
C
          DO 303 I=1,NH
          DO 303 J=1,NH
          IF (HNM1(I,J).GE.0) GOTO 303
          HNM1(I,J)=0.0
303       CONTINUE
C
C******_COMPUTE RMS DEVIATION
C
          ERR=0
C
          DO 304 I=1,NH
          DO 304 J=1,NH
304       ERR=ERR+(HNM1(I,J)-HN(I,J)+H(I,J))**2
C
          NH2=NH*NH
          ERR=(ERR/NH2)**.5
C
C******_OUTPUT
C
          WRITE (31 I11) HNM1
          WRITE(20,6)I11,ERR
300       TYPE 6,I11,ERR
          END
```

```
      SUBROUTINE CON2(IN,FILT,INX,INY,INXM,INYM,FILTX,FILTY,
     *                FILTXM,FILTYM,OUT,OUTX,OUTY,OUTXM,OUTYM,
     *                PEAKX,PEAKY,ERR)
      INTEGER INX,INY,FILTX,FILTY,OUTX,OUTY,HYE,HYE,PEAKX,PEAKY
      INTEGER INXM,INYM,FILTXM,FILTYM,OUTXM,OUTYM,GROW,GCOL
      REAL IN,FILT,OUT
      DIMENSION IN(INXM,INYM),FILT(FILTXM,FILTYM),OUT(OUTXM,OUTYM)
      DIMENSION MXGROW(64),MXGCOL(64),MNROW(64),MXROW(64),
     *          MNCOL(64),MXCOL(64)
C     ARRAY IN CONTAINS INPUT DATA, ARRAY FILT CONTAINS FILTER POINTS
C     ARRAY OUT CONTAINS RESULT OF (IN*FILT)
C     INX,INY,FILTX,FILTY ARE X AND Y DIMENSIONS OF INPUT AND FILTER
C     PEAKX,PEAKY ARE ROW,COLUMN INDICES OF THE PEAK OF FILT(I,J)
C     ROWS AND COLUMNS ARE ALWAYS NUMBERED FROM 1 UPWARD
C     OUTX,OUTY ARE THE DIMENSIONS OF THE OUTPUT DATA....
C     INXM,INYM,FILTXM,FILTYM,OUTXM,OUTYM REPRESENT DIMENSIONS OF THE
C     RESPECTIVE ARRAYS IN THE CALLING PROGRAM.  THESE MUST BE THE
C     DIMENSIONS FROM THE MAIN PROGRAM DIMENSION STATEMENT
C     .. THE M SUFFIX IMPLIES MAIN DIMENSIONS ..
C     .. WHILE THE CORRESPONDING VARIABLE IS THE VALID DATA DIMENSION
..
C
C     ASSIGN OUTPUT DIMENSIONS (SAME AS INPUT)
      OUTX=INX
      OUTY=INY
C     COMPUTE LIMITS OF SUMMATIONS
      DO 40 L=PEAKX,OUTX+PEAKX
```

```
      MNROW(L)=MAX0(1,(L-FILTX+1))
      MXROW(L)=MIN0(L,INX)
40    MXGROW(L)=MIN0(L,FILTX)
      DO 30 L=PEAKY,OUTY+PEAKY
      MNCOL(L)=MAX0(1,(L-FILTY+1))
      MXCOL(L)=MIN0(L,INY)
30    MXGCOL(L)=MIN0(L,FILTY)
C     PERFORM NONEXPANDING CONVOLUTION
      MX=PEAKX-1
      MY=PEAKY-1
      DO 10 LROW=PEAKX,OUTX+PEAKX
      GROW=MXGROW(LROW)
      DO 10 IROW=MNROW(LROW),MXROW(LROW)
      DO 20 LCOL=PEAKY,OUTY+PEAKY
      GCOL=MXGCOL(LCOL)
      DO 20   ICOL=MNCOL(LCOL),MXCOL(LCOL)
      OUT(LROW-MX,LCOL-MY)=OUT(LROW-MX,LCOL-MY)
                                    +IN(IROW,ICOL)*FILT(GROW,GCOL)
20    GCOL=GCOL-1
10    GROW=GROW-1
      ERR=0
      RETURN
      END
```

LIST OF REFERENCES

Andrews, H.C. and Hunt B.R. (1977), Digital Image Restoration, (Englewood Cliffs: Prentice Hall, 1977).

Billingsley, F.C. (1979), "Noise Considerations in Digital Image Processing Hardware," Topics in Applied Physics, Vol. 6, Picture Processing and Digital Filtering, ed. by T.S. Huang. (New York: Springer-Verlag, 1979), pp. 249-280.

Bracewell, R.N. (1978) The Fourier Transform and Its Applications, (New York: McGraw-Hill, 1978).

Frieden, B.R. (1979), "Image Enhancement and Restoration," Topics in Applied Physics, Vol. 6, Picture Processing and Digital Filtering, ed. by T.S. Huang. (New York: Springer-Verlag, 1979), pp. 179-246.

Gaskill, J.D. (1978), Linear Systems, Fourier Transforms, and Optics. (New York: John Wiley & Sons, 1978)

Hamming, R.W. (1962), Numerical Methods for Scientists and Engineers, (New York: McGraw-Hill, 1962).

Hill, N.R. (1973), "Deconvolution for Resolution Enhancement," (Unpublished master's thesis, Dept. of Physics, University of New Orleans, 1973).

Hill, N.R. and Ioup, G.E. (1976), "Convergence of the van Cittert iterative method of deconvolution," J. Opt. Soc. Am., Vol. 66, No. 5, (1976) pp. 487-489.

Ioup, G.E. (1968), Analysis of Low Energy Atomic and Molecular Collisions: Semiclassical Elastic Scattering Calculations and Deconvolution of Data, Ph. D. dissertation, University of Florida (1968).

Ioup, G.E. (1979), private communications.

Knuth D.E. (1969), The Art of Computer Programming, Vol. 2, (Reading, Mass.: Addison-Wesley, 1969).

Morrison, J.D. (1963), "On the Optimum Use of Ionization Efficiency Data," J. Chem. Phys., Vol. 39, No. 1 (1963), pp. 200-207.

Oppenheim, A.V, and Schafer, R.W. (1975), Digital Signal Processing, (Englewood Cliffs: Prentice-Hall, 1975).

Whitehorn, K.A. (1980), "A Study of Derivative Filters Using the Discrete Fourier Transform," (Unpublished master's thesis, Dept. of Physics, University of New Orleans, 1980).

VITA

Mark Alan Whitehorn was born in [redacted], on [redacted], to Jamie Nell Whitehorn and F. Sherman Whitehorn. He graduated from Bonnabel High School in May 1973 and entered the University of New Orleans in the fall of that year. He received the degree of Bachelor of Science in physics in May 1977. On August 6, 1977, he married Kathleen [redacted], who now holds the degree of Master of Science in physics.

From May of 1977 to the fall of 1978 he worked as a seismologist for Geophysical Service Incorporated. He left GSI to accept a position as Systems Designer with the Computer Research Center at the University of New Orleans. He also entered the graduate school at UNO in the fall of 1978.

Following graduation he will be employed as a Scientific Programmer by the Lockheed Missiles and Space Corporation in Sunnyvale, California.